VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MOBILE IPV6 V PROSTŘEDÍ OPNET MODELER

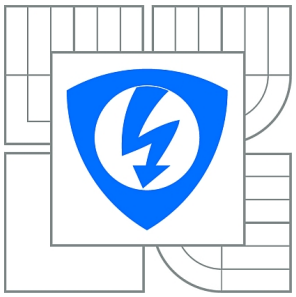
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MARTIN ŽÁČEK
AUTHOR

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MOBILE IPV6 V PROSTŘEDÍ OPNET MODELER

MOBILE IPV6 IN OPNET MODELER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

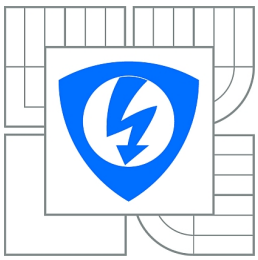
AUTOR PRÁCE — Bc. MARTIN ŽÁČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL SKOŘEPA

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Žáček	***ID:***	77945
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Mobile IPv6 v prostředí OPNET Modeler

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku Mobile IPv6. Zvláštní pozornost věnujte procesům zajišťujícím úspěšné předání mobilního uzlu mezi sítěmi, a to pro různá schémata tohoto předání, jako např. FMIPv6, HMIPv6, F-HMIPv6 atd. Detailně prostudujte zprávovou výměnu mezi entitami protokolu Mobile IPv6, ke které v průběhu předání dochází. V simulačním prostředí OPNET Modeler vytvořte funkční simulaci sítě implementující protokol Mobile IPv6.
V editoru uzlu, resp. v editoru procesů, v prostředí OPNET Modeler analyzujte procesy vedoucí k zajištění handoveru v MIPv6. Upravte tyto procesy tak, aby byl handover zajištěn schématem FMIPv6. Pomocí simulace pak ověřte, jaký vliv na přenášená data má toto schéma oproti schématu původnímu.

DOPORUČENÁ LITERATURA:

[1] KOODLI, R. S., PERKINS, C. E. Mobile Inter-networking with IPv6: Concepts, Principles and Practices. Wiley-Interscience, 2007. 365 pages. ISBN 978-0471681656
[2] JOHNSON, D. B., PERKINS, C. E., ARKKO, J. Mobility support in IPv6. The Internet Engineering Force Task RFC 3775, June 2004
[3] PÉREZ COSTA, X., SCHMITZ, R., HARTENSTEIN, H., LEIBSCH, M. A MIPv6, FMIPv6 and HMIPv6 Handover Latency Study: Analytical Approach. In Proceedings of the IST Mobile and Wireless Communication Summit, June 2002, p. 100 - 105

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Michal Skořepa

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ANOTACE

Diplomová práce shrnuje poznatky o podpoře mobility v IPv4 sítích a detailně prozkoumává možnosti mobility v IPv6 sítích. Jsou zde popsány funkce a topologie protokolů MIPv6 (RFC 3775), FMIPv6 (RFC 5568), HMIPv6 (RFC 5380) a F-HMIPv6. Dále práce obsahuje návrh simulace MIPv6 v prostředí OPNET Modeler a diskuzi výsledků k této simulaci. V neposlední řade, byla provedena analýza zdrojových kódů programu OPNET, pro účely návrhu vlastního protokolu pro podporu mobility, dle FMIPv6. V implementaci FMIPv6 se podařila vyřešit detekce událostí z linkové vrstvy, které slouží ke spouštění vlastního protokolu a byly implementovány dvě zprávy RtSolPr a PrRtAdv. Poznatky z této práce mohou sloužit k dokončení implementace protokolu FMIPv6 v prostředí OPNET, ale také mohou být využity obecně jako metodika práce při návrhu nových typů zpráv přenášených na síťové vrstvě v IPv6 síti.

Klíčová slova: MIPv6, FMIPv6, OPNET, mobilita, simulace, návrh, protokol

ABSTRACT

Master's thesis includes knowledges about mobility support in IPv4 networks and studies abilities for mobility support in IPv6 networks. Describes protocols for mobility support, their functions and topologies like MIPv6 (RFC 3775), FMIPv6 (RFC 5568), HMIPv6 (RFC 5380) and F-HMIPv6. The thesis contains a design of the simulation MIPv6 in the OPNET Modeler program and the measured results are explained. There are analysis of source code for mobility support in MIPv6 too, which were used for design a new protocol according to FMIPv6 in the OPNET. The following parts of the proposal have been successfully resolved. The link-layer events detection, which triggers FMIPv6 protocol and implementation of two new types of messages, RtSolPr and PrRtAdv. Knowledges from this thesis could be used for next development of FMIPv6 in the OPNET. But could be used for generaly development new protocols working at network-layer too.

Keywords: MIPv6, FMIPv6, OPNET, mobility, simulation, design, protocol

ŽÁČEK, M. *Mobile IPv6 v prostředí OPNET Modeler.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 103 s. Vedoucí diplomové práce Ing. Michal Skořepa.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma *Mobile IPv6 v prostředí OPNET Modeler* jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 21.5.2010

.....

podpis autora

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat několika lidem, kteří mě podpořili při psaní této práce. Především mé rodině za trpělivost, protože v období psaní této práce, to se mnou neměli jednoduché. Pak také Láďovi Mikulicovi za pomoc při práci s OPNETem. Ing. Jiřímu Hoškovi za poskytnutí materiálů a za podporu. A nakonec svému vedoucímu Ing. Michalu Skořepovi, za optimistický přístup k řešení problémů, kterého se mě samotnému někdy nedostávalo.

OBSAH

OBSAH **1**

SEZNAM OBRÁZKŮ **4**

ÚVOD **5**

I) TEORETICKÁ ČÁST **6**

1 PODPORA MOBILITY V IPV4 ..... **6**

1.1 IMPLEMENTACE MOBILITY UVNITŘ IPV4 6
1.2 POJMY SPOJENÉ S PODPOROU MOBILITY V IPV4 6
1.3 TECHNIKY POUŽÍVANÉ V IPV4 SOUVISEJÍCÍ S PODPOROU MOBILITY 7
1.3.1 Objevení agenta (Agent discovery) *7*
1.3.2 Registrace *7*
1.3.3 Tunelování *7*
1.3.4 Optimalizace *7*
1.4 POPIS FUNKCE MIPV4 8
1.5 NEVÝHODY MIPV4 9

2 ZÁKLADNÍ PODPORA MOBILITY V IPV6 (MIPV6) **9**

2.1 IMPLEMENTACE MOBILITY UVNITŘ IPV6 9
2.2 POPIS TECHNIK PRO PODPORU MOBILITY POUŽÍVANÝCH V IPV6 10
2.2.1 Autokonfigurace *10*
2.2.2 Ohlášení směrovače (Router advertisement = RA) *10*
2.2.3 Objevování sousedů (Neighbor Discovery) *11*
2.2.4 Objevování domácích agentů (Home Agent Discovery Mechanism) *11*
2.2.5 Ověření zpětného směrování (Return Routability Procedure = RRP) *12*
2.3 ZPRÁVY A JEJICH HLAVIČKY DEFINOVANÉ PRO PODPORU MOBILITY V IPV6 14
2.3.1 Žádost o obnovení vazby (binding refresh request) *14*
2.3.2 HoTI, CoTI, HoT, CoT (zprávy 1 až 4 dle tabulky) *15*
2.3.3 Aktualizace vazby (binding update = BU) *15*
2.3.4 Potvrzení vazby (binding acknowledgement = BAck) *15*
2.3.5 Chyba vazby (binding error) *16*
2.4 TOPOLOGIE MIPV6 16
2.4.1 Korespondent (CN) *16*
2.4.2 Domácí agent (HA) *17*
2.4.3 Mobilní uživatel (MN) *17*
2.4.4 Popis pohybu MN v topologii MIPv6 ..... *17*
2.5 KOMUNIKACE MEZI KORESPONDENTEM A MOBILNÍM UŽIVATELEM (BEZ OPTIMALIZACE) 18
2.5.1 Popis komunikace *18*
2.6 KOMUNIKACE MEZI KORESPONDENTEM A MOBILNÍM UŽIVATELEM (S OPTIMALIZACÍ) 20
2.6.1 Popis komunikace *20*
2.7 SHRNUTÍ K MIPV6 22

3 RYCHLÝ HANDOVER V MIPV6 [FMIPV6] **22**

3.1 ZPRÁVY A JEJICH HLAVIČKY DEFINOVANÉ V RÁMCI FMIPV6 23
3.2 POPIS PROCESU KOMUNIKACE PREDIKTIVNÍHO REŽIMU FMIPVV6 25

3.2.1 Popis komunikace ... *25*
3.3 Popis procesu komunikace reaktivního režimu FMIPv6 ... 28
3.3.1 Popis komunikace ... *28*
3.4 Shrnutí k FMIPv6 ... 30
4 HIERARCHICKÁ MOBILITA [HMIPV6] ... 30
4.1 Důležité hlavičky definované pro HMIPv6 ... 30
4.1.1 Objevení MAPu ... *30*
4.1.2 Formování adresy RCoA a LCoA ... *31*
4.2 Topologie HMIPv6 ... 32
4.3 Popis procesu komunikace v HMIPv6 ... 32
4.4 Shrnutí k HMIPv6 ... 35
5 RYCHLÝ HANDOVER V HIERARCHICKÉ SÍTI [F-HMIPV6] ... 35
5.1 Implementace FMIPv6 do HMIPv6 ... 35
5.2 Zprávy a jejich hlavičky definované v rámci F-HMIPv6 ... 36
5.3 Popis síťové komunikace v doméně F-HMIPv6 ... 37
5.4 Shrnutí k F-HMIPv6 ... 38
6 ÚVOD DO VÝVOJOVÉHO PROSTŘEDÍ OPNET ... 40
6.1 Editor projektu ... 40
6.2 Editor uzlu ... 40
6.2.1 Komunikace mezi uzly a procesy ... *41*
6.3 Procesní model ... 42
6.3.1 Stavové proměnné ... *43*
6.3.2 Blok hlaviček ... *43*
6.3.3 Blok funkcí ... *43*
II) PRAKTICKÁ ČÁST ... 44
7 SIMULACE MIPV6 V PROGRAMU OPNET ... 44
7.1 Zadání ... 44
7.2 Řešení ... 44
7.2.1 Vytváření topologie ... *44*
7.2.2 Konfigurace prvků topologie ... *46*
7.2.3 Nastavení simulace ... *54*
7.3 Výsledky simulace ... 54
8 ANALÝZA ZDROJOVÝCH KÓDŮ MIPV6 ... 58
8.1 Mipv6_mgr model procesů ... 58
8.1.1 Mipv6_mgr – stav init ... *60*
8.1.2 Mipv6_mgr – stav idle ... *63*
8.2 Mipv6_mn model procesů ... 69
8.2.1 Dostupné datové struktury a proměnné ... *72*
8.2.2 Popisy jednotlivých stavů a vybraných funkcí ... *72*
8.3 Wlan_mac ... 74
8.4 Modely pro přeposílání paketů ... 75
8.4.1 ip_rte_central_cpu model ... *76*

9 IMPLEMENTACE FMIPV6 78

9.1 DETEKCE HANDOVERU A VÝMĚNA INFORMACÍ MEZI L2 A L3 79

9.1.1 Implementace událostí na linkové vrstvě 79

9.1.2 Detekce přerušení na síťové vrstvě 83

9.2 IMPLEMENTACE ZPRÁV RTSOLPR A PRRTADV 85

9.2.1 Funkce pro odeslání RtSolPr zprávy 85

9.2.2 Modifikace funkcí zpracovávající rozšiřující hlavičky IPv6 paketů 88

9.2.3 Modifikace podpory mobility ze strany routerů FN1 a FN2 91

9.2.4 Modifikace funkcí v podpůrné knihovně mipv6_sup.ex.c 92

9.2.5 Zpracování zprávy 93

9.2.6 Zpráva PrRtAdv 96

9.3 NÁVRH ŘEŠENÍ DALŠÍCH ČÁSTÍ FMIPV6 97

10 ZÁVĚR 99

11 LITERATURA 101

12 SEZNAM ZKRATEK, VELIČIN A SYMBOLŮ 103

SEZNAM OBRÁZKŮ

Obr. 1.1: Registrace MN u HA: a) CoA přes FA, b) coalokovaná CoA přímo 8
Obr. 2.1: Zpráva Ohlášení směrovače 10
Obr. 2.2: Průběh RRP procedury 12
Obr. 2.3: Rozšiřující hlavička pro podporu mobility v IPv6 14
Obr. 2.4: Hlavička zprávy BU 15
Obr. 2.5: Hlavička zprávy BAck 16
Obr. 2.6: Topologie MIPv6 17
Obr. 2.7: Zprávová výměna při komunikaci CN-MN v základním MIPv6 19
Obr. 2.8: Zprávová výměna při komunikaci CN-MN v optimalizovaném MIPv6 21
Obr. 3.1: Topologie FMIPVv6 – počátek pohybu MN 22
Obr. 3.2: Zprávová výměna při komunikaci MN s prvky PAR a NAR v rámci prediktivního FMIPVv6 27
Obr. 3.3: Zprávová výměna při komunikaci MN s prvky PAR a NAR v rámci reaktivního FMIPVv6 29
Obr. 4.1: Hlavička zprávy Objevení MAPu. 31
Obr. 4.2: Hlavička pro podporu mobility (upravená pro HMIPv6). 31
Obr 4.3: Topologie HMIPv6 33
Obr. 4.4: Zprávová výměna mezi prvky sítě při HMIPv6 34
Obr. 5.1: Implementace FMIPv6 v rámci HMIPv6 36
Obr. 5.2: Zprávová výměna mezi prvky topologie F-HMIPv6 39
Obr. 6.1: Editor uzlu – ukázka propojení jednotlivých bloků 41
Obr. 6.2: Procesní model – ukázka propojován stavů 43
Obr. 7.1: Topologie MIPv6 simulovaná v prostředí Opnet. 45
Obr. 7.2: Nastavení aplikace FTP 47
Obr. 7.3: Nastavení profilu FTP 47
Obr. 7.4: Nastavení parametrů mobility MN 49
Obr. 7.5: Nastavení IPv6 parametrů MN 50
Obr. 7.6: Nastavení parametrů bezdrátové sítě pro MN 51
Obr. 7.7: Nastavení parametrů mobility CN 52
Obr. 7.8: Nastavení parametrů ohlášení směrovače 53
Obr. 7.9: Výsledek1 55
Obr. 7.10: Výsledek2 56
Obr. 7.11: Výsledek3 57
Obr. 8.1: Stavový diagram *mipv6_mgr* modelu 59
Obr. 8.2: Diagram funkce mipv6_attributes_read – část 1 61
Obr. 8.3: Diagram funkce *mipv6_attributes_read()* – část 2 62
Obr. 8.4: Diagram funkce mipv6_attributes_read() – část 3 63
Obr. 8.5: Diagram zprávy BU, tak jak je zpracována funkcí *mipv6_mgr_mob_msg_send()* – část1 67
Obr. 8.6: Diagram zprávy BU, tak jak je zpracována funkcí *mipv6_mgr_mob_msg_send()* – část2 68
Obr. 8.7: Stavový diagram procesního modelu mipv6_mn 70
Obr. 8.8: Procesní model *wlan_mac* 74
Obr. 8.9: Wi-fi rámec 75
Obr. 8.10: Schéma odesílání paketu z mipv6_mn 77
Obr. 9.1: Model simulace pro FMIPv6 79
Obr. 9.2: Přidání nových atributů pro zajištění podpory FMIPv6 u AP 91

ÚVOD

Zamyslíme-li se nad problémem podpory mobility v počítačových sítích, tak zjistíme, že v poskytování mobility jsme už ušli pěkný kus cesty. Máme telekomunikační mobilní sítě, díky kterým je možné spojit se mobilním telefonem s kýmkoliv po celém světě. Přesto však telekomunikační mobilní sítě neposkytují takové možnosti, jaké by mohly poskytovat mobilní počítačové sítě. Prioritním úkolem telekomunikačních mobilních sítí je poskytnutí hovorových služeb, kdežto počítačové sítě jsou datového charakteru a jejich použitelnost má mnohem větší záběr (přenos velkých datových objemů, internet, email, streeming, VoIP, a mnohé další), zatím však bohužel jen na lokální úrovni poskytovatele této služby. Provozovatelé telekomunikačních mobilních sítí se snaží i o zavedení nových datových služeb (internet v mobilu), ty však mají mnohem větší požadavky na datovou propustnost než hovorové služby, a tak je jejich zavádění pozvolné, protože je drahé.

A jaký je vývoj v podpoře mobility v počítačových sítích? Bohužel v původním návrhu IPv4 nebylo uvažováno s mobilitou uživatelů. Zařízení, které je připojené k LAN, potřebuje znát svoji IP adresu a adresu výchozí brány. Tyto nastavení jsou dnes povětšinou přidělovány automaticky DHCP serverem na základě MAC adresy zařízení. Má-li tyto parametry uživatel přidělen, může využívat služeb, které jsou počítačovou sítí a provozovatelem poskytovány, ne však za hranicemi této sítě. Z pohledu mobilního uživatele nevidíme problém, ale podíváme-li se výše, vyvstanou otázky: „Je síťový administrátor schopen reagovat rychle na nové požadavky mobilních uživatelů v jeho síti? Jak bude administrátor provádět autentizaci těchto uživatelů?“

Odpovědi na tyto otázky nejsou pozitivní, ale řešení přeci jen existuje. Velmi slibně vypadá zahrnutí mobility do IPv6 sítí. V IPv6 jsou zahrnuty mechanizmy, které umožňují příchozí, do této chvíle neznáme stanici, nakonfigurovat si automaticky připojení k této síti. A výzkum šel ještě dále. IPv6 mechanizmy byly vylepšeny technikami rychlého handoveru (FMIPVv6), zavedením hierarchie v mobilních sítích (HMIPv6) a poskytování rychlého handoveru v HMIP (F-HMIPv6). MIPv6 a HMIPv6 jsou dnes již připraveny k nasazení, protože jsou specifikovány v RFC dokumentech. Další možnosti jsou ve stádiu výzkumu.

V mé práci Vás obeznámím s tím, jak fungují MIPv6, HMIPv6 a F-HMIPv6 sítě, co poskytují uživateli. Bude popsána tvorba simulace MIPv6 protokolu a bude navržena část protokolu FMIPv6. Práce je rozdělena na dvě části. Teoretická pojednává o možnostech poskytování mobility v počítačových sítích a praktická část se zabývá analýzou řešení protokolu FMIPv6 v prostředí OPNET Modeler, implementací protokolu FMIPv6 a simulací stávajícího řešení mobility, která je zajištěná protokolem MIPv6. Závěr shrnuje dosažené výsledky.

I) TEORETICKÁ ČÁST

1 PODPORA MOBILITY V IPv4

1.1 Implementace mobility uvnitř IPv4

Protokol IPv4 ve svém základu nedisponuje podporou mobility, protože na to není topologicky připraven. Routery, které udržují informace o směrování uživatelů, nejsou konstruované na velmi rychlé změny routovacích tabulek, jsou konstruovány na rychlé přeposílání paketů, dle routovacích tabulek [4]. Avšak díky rozvoji internetu a novým požadavkům, bylo nutné, podporu mobility zajistit. V RFC 3344 z roku 2002 je popsáno, jak lze v IPv4 sítích mobility dosáhnout [1]. Definuje nové prvky v topologii internetu, techniky přístupu k mobilním uživatelům, popisuje jejich registraci v cizí síti, a také zprávovou výměnu mezi těmito prvky topologie. V souvislosti s podporou mobility, zde definovanou, vzniká zkratka MIPv4 (Mobile Internet Protocol version 4).

1.2 Pojmy spojené s podporou mobility v IPv4

V rámci MIPv4 jsou definovány tyto základní pojmy [4, 11]:

- **Mobilní uživatel** (MN = Mobil Node) je uživatel, který se pohybuje mezi několika sítěmi.
- **Korespondent** (CN = Corespondent Node) je uživatel, který navazuje komunikaci s MN.
- **Domácí adresa** (Home Adress) je adresa MN při pobytu v domácí síti (HN = Home Network).
- **CoA** (Care-of Adress) je adresa, kterou MN získá při přechodu do cizí sítě (FN = Foreign Network). Tato adresa se použije pro mobilního uživatele. První variantou je, že CoA ukazuje na agenta v cizí síti (FA), druhou možností, že se jedná o Co-CoA. V tomto případě není potřeba v cizí síti FA a CoA získá přímo mobilní uživatel.
- **Domácí agent** (HA = Home Agent) je prvek v domácí síti, který zajišťuje dostupnost MN.
- **Agent cizí sítě** (FA = Foreign Agent) je prvek v síti FN (tedy v síti kde se MN aktuálně nachází).

[1] RFC (Request For Comment) jsou základními technickými dokumenty, které se zabývají nejrůznějšími aspekty fungování Internetu. Nejsou zákonnými ustanoveními, ani standardními normami, ale přesto jsou ctěny a dodržovány. RFC píší různí experti. Za označením RFC následuje číslo (např.: RFC 3344), a řeší určitou problematiku fungování internetu. Seznam lze nalézt na: http://www.ietf.org/rfc.html.

1.3 Techniky používané v IPv4 související s podporou mobility

1.3.1 Objevení agenta (Agent discovery)

Vycestuje-li mobilní uživatel mimo domovskou síť, potřebuje v cizí síti získat adresu (Care-of Adress). Ta může být přidělena dvěma způsoby, a sice technikou agent advertisement nebo agent solicitation [11].

Agent advertisement

Tato technika umožňuje ohlásit přítomnost FA nebo HA v síti, do které MN vycestoval. Router v cizí síti, který zastává tuto funkci, vysílá v pravidelných intervalech na multicastovou adresu oznámení (advertisement). Jedná se o ICMP zprávu (dle RFC 1256) rozšířenou o podporu mobility. Prostřednictvím této zprávy se MN dozví, kdo bude jeho FA agent.

Agent Solicitation

Agent solicitation je aktivní metodou. MN vyšle sám požadavek na přidělení CoA. Tato zpráva je definovaná v RFC 1256. Při velkém počtu mobilních uživatelů by bylo riziko, že se vysíláním solicitation zpráv zahltí síť, proto je omezeno její vysílání. Typicky třikrát do minuty, s tím, že při neohlášení FA, se interval solicitation zpráv exponenciálně snižuje.

1.3.2 Registrace

MN získá CoA a následuje registrace mezi HA a MN (v případě coalokované CoA), jak ukazuje obrázek 1.1a, respektive přenos registrace mezi MN-FA-HA (v případě že Coa bude náležet FA), jak ukazuje obrázek 1.1b.

Procesu registrace se říká „mobility binding“ [4], tedy tzv. vazba mobility. Hlavička zprávy „Žádost registrace“ (Registration request) obsahuje bity, kterými informuje HA resp. FA o možnostech MN. Domovskou IP adresu MN, IP Domácího agenta, CoA, identifikaci (slouží pro zabezpečení) a rozšiřující informace. Na zprávu „Žádost registrace“ odpovídá HA resp. FA-HA zprávou „Potvrzení registrace“ (Registration reply). Po procesu registrace si HA nastaví CoA mobilního uživatele. Vazba se poté pravidelně aktualizuje, dle nastavení v hlavičce.

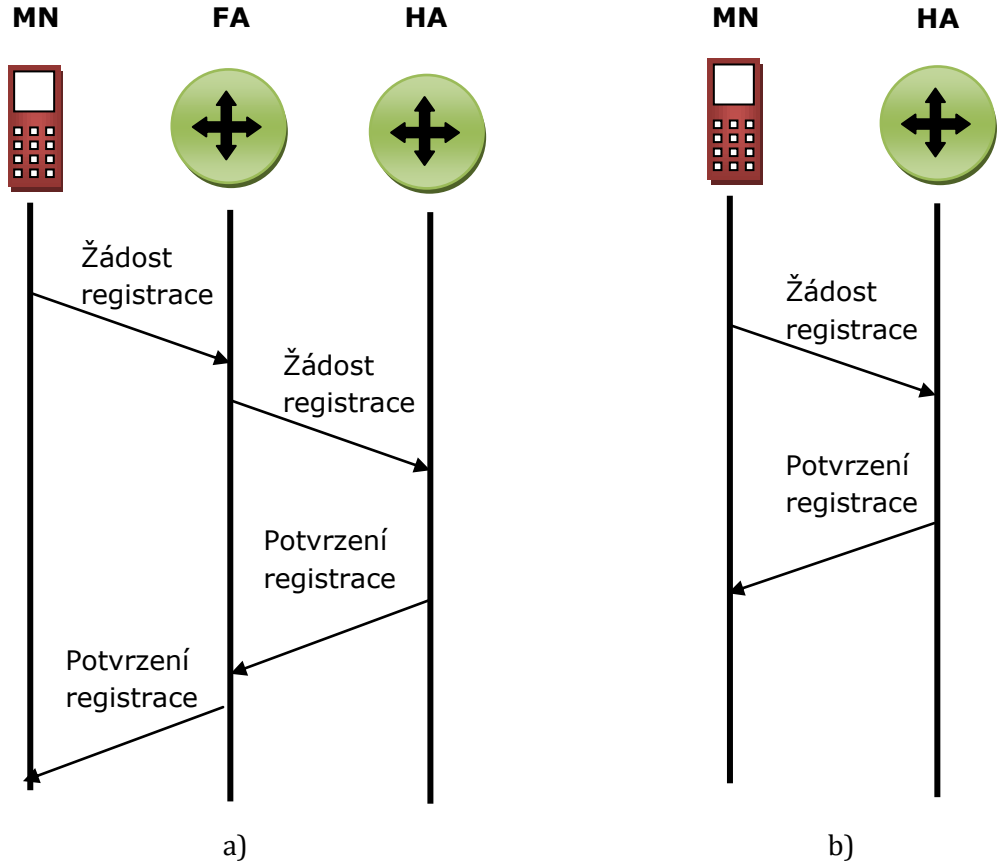
1.3.3 Tunelování

Způsob, jakým se přenášejí data mezi HA a MN. Data putující od korespondenta jsou v HA zabalena a je k nim připojena hlavička, která oznamuje, že data mají být dále směrována na CoA.

1.3.4 Optimalizace

Základní myšlenka mobility v IPv4 není příliš efektivní. Uvažujme případ, že spolu chtějí komunikovat dvě zařízení, které jsou od sebe topologicky blízko. Přesto dle základní myšlenky MIPv4 budou muset být veškerá data posílána přes HA. Tomuto neefektivnímu směrování se říká trojúhelníkové směrování (triangular routing) [4].

Situaci lze vyřešit tak, že CN bude vědět o poloze MN a bude mu data posílat přímo, ne přes HA. Také v souvislosti s bezpečností, musela být zavedena nová rozšíření. Jedná se o čtyři nové zprávy (originální pojmenování jsou: „Binding request“, „Binding Update“, „Binding Acknowledgement“, „Binding Warning“) a z hlediska struktury, přibylo na straně CN rozšíření v podobě „Binding Cache“.



Obr. 1.1: Registrace MN u HA: a) CoA přes FA, b) coalokovaná CoA přímo

1.4 Popis funkce MIPv4

Vycestuje-li uživatel MN mimo domácí síť (HN), zažádá si v cizí síti (FN) o přidělení adresy (CoA), skrze DHCP server a informuje o tom domácího agenta (HA). Pomocí zprávy (Registration Request). Na tuto zprávu HA odpoví (Registration Reply). Od této chvíle plní HA prostředníka v komunikaci mezi korespondentem (CN) a HN. Zachytí všechny pakety přicházející od CN a s využitím Proxy ARP je přepošle tunelováním na CoA. Tyto pakety zachytí agent cizí sítě (FA). Z paketů odstraní přebytečnou hlavičku, která slouží pro směrování paketů do cizí sítě (FN) a pošle MN (který se nachází uvnitř lokální sítě FN). Při posílání paketů zpět k CN, jsou pakety posílány přes FA přímo k cíli.

1.5 Nevýhody MIPv4

Registrace MN v rámci MIPv4 zabírá dlouhý čas [4], což způsobuje prodlevu při handoveru. Pakliže se MN pohybuje mezi sítěmi při probíhající komunikaci tak rychle, že se nestíhá vyřídit registrace MN do FN, dochází ke ztrátě paketů. Dále také časté vysílání zpráv (Agent Advertisement a Registration), které se zvětšuje se zvětšujícím se počtem mobilních uživatelů, zatěžuje FN. V souvislosti s uvedenými nevýhodami, jsou navrhovány nové metody handoveru, které se snaží tyto nevýhody odstranit. Jedná se o techniky optimalizace směrování (routing optimization technique), "anticipation" techniku, "hierarchical" techniku.

2 ZÁKLADNÍ PODPORA MOBILITY V IPv6 (MIPv6)

IP verze 6 je pokračovatelem IP verze 4. Přestože má nesčetné výhody a v budoucnu bude nutné tímto protokolem nahradit stávající IPv4, je dnes jeho nástup pozvolný. Nasazování IPv6 bude probíhat od shora (kořenové routery) dolů (uživatel). U uživatele je podpora IPv6 už od roku 2000[2], provozovatelé internetu však přecházejí pouze obtížně. Příčinou je obtížnější konfigurace protokolu a nutná podpora ze strany hardwaru.

„První implementace proběhla v síti CESNET už před deseti lety, v rámci NIX.CZ se k ní přistoupilo až v roce 2003. K dnešnímu datu (červenec 2009) je pro Českou republiku přiděleno 52 IPv6 prefixů. Přibližně 36 % sítí připojených do NIXu používá zároveň IPv6 peering. Velká část IPv6 sítí je ale stále experimentální a nevyužívají je uživatelé. Největšími sítěmi (dle provozu) používajícími IPv6 jsou Casablanca, CESNET, CoolHousing a CZ.NIC.“ [zdroj: CESNET http://www.cesnet.cz/sdruzeni/napsali-o-nas/2009/06/20090609_Root.html].

Mezi hlavní přednosti IPv6 protokolu patří [8]

- **větší adresní prostor – 128 bitů místo 32 bitů v IPv4,**
- **autokonfigurace stanic,**
- **podpora mobility,**
- **zajištění QoS,**
- **objevování sousedů,**
- **efektivnější návrh hlavičky.**

2.1 Implementace mobility uvnitř IPv6

Následuje stejné základy podpory mobility, které byly definovány pro IPv4. Mechanismy pro podporu mobility byly sice vymyšleny i pro starší protokol, ale z hlediska efektivity a nabízených

[2] V roce 2000 byla implementovaná podpora IPv6 v systémech Linux a od roku 2002 v systému Windows XP SP1. Novější operační systémy mají podporu IPv6 standardně obsaženu

možností za IPv6 zřetelně zaostávají. Používá se pro ně označení Mobile IPv6 (MIPv6). Podpora mobility pro IPv6 je standardizována v RFC 3775 z roku 2004. V rámci IPv6 byla definována podpora mobility tak, aby umožňovala zařízení obstarat si IP adresu i při změnách lokace a zároveň mít vždy plně funkční připojení. Tento mechanismus pracuje na síťové vrstvě TCP/IP modelu. Pro vyšší vrstvy modelu, je neviditelný. Podpora mobility ovlivňuje, jak budou pakety routovány, ale přitom neovlivňuje routovací protokoly [4].

2.2 Popis technik pro podporu mobility používaných v IPv6

2.2.1 Autokonfigurace

Autokonfigurace stanic v síti je proces, který nově příchozí stanici do sítě (PC se síťovou kartou, notebook s wi-fi kartou…) nastaví parametry Ip adresu, masku podsítě, výchozí adresu brány, DNS servery tak, aby mohla využít připojení k síti LAN (případně i do internetu) [8].

Konfigurace těchto parametrů může být prováděna *stavově* nebo *bezstavově*. V IPv6 protokolu jsou povoleny obě metody. Stavová metoda v IPv6 je založena na fungování DHCPv6 serveru. Uživatel, který přijde do sítě, si zažádá o přidělení parametrů pomocí broadcastové zprávy. Bezstavová konfigurace funguje na principu, že existují v síti LAN směrovače, které vědí o nově příchozích stanicích do sítě a o rozesílání parametrů se starají tyto směrovače.

V souvislosti s mobilitou IPv6, každé mobilní zařízení, které se octne v „cizí“ síti, provede autokonfiguraci. Získá tedy adresu z této sítě (tzv. *CoA*), kterou použije pro komunikaci se svým domovským agentem. Domovský agent bude tedy za každé situace vědět, na jaké adrese je mobilní uživatel dostupný.

2.2.2 Ohlášení směrovače (Router advertisement = RA)

Zajišťuje fungování bezstavové autokonfigurace. Posílá je v náhodných intervalech každý směrovač, a to do všech sítí, k nimž je připojen. Přestávka mezi ohlášeními je náhodná. Po obdržení ohlášení směrovače vědí připojené počítače, v jaké síti se to octly, jak se zde komunikuje a kdo je implicitní směrovač. Obrázek 2.1 znázorňuje hlavičku zprávy RA [1, 8].

Obr. 2.1: Zpráva Ohlášení směrovače

Význam jednotlivý částí hlavičky je tento (více informací v [8]):

Typ = 134, **Kód** = 0,

bity **M**, **O** vypovídají o použití DHCPv6,

bit **H** vypovídá o použití směrovače jako Domácího agenta,

Trvání dosažitelnosti souvisí s detekcí dosažitelnosti sousedů, jedná se o časový údaj.

2.2.3 Objevování sousedů (Neighbor Discovery)

Jedním z dobře známých problémů počítačových sítí je zjištění linkové adresy partnera (MAC adresa). Počítač potřebuje poslat někomu data, zná jeho IP adresu, podle ní také ví, že spolu sídlí v jedné lokální síti. Aby mu však mohl odeslat paket, potřebuje znát právě cílovou MAC adresu.

V IPv4 je pro zjištění IP-MAC použit ARP[3]. V IPv6 se mechanismus zjišťování ardes IP-MAC definuje jinak. Jedná se o techniku Objevování sousedů (Neighbor Discovery). Tato technika zajišťuje [8]:

- Zjišťování MAC adres prvků v lokální síti,
- hledání směrovačů,
- aktualizace MAC adres prvků v LAN,
- přesměrování,
- zjišťování parametrů potřebných pro konfiguraci prvků v LAN (IP, adresa brány, maska,…),
- ověřování dosažitelnosti sousedů,
- detekce duplicitních adres.

Konkrétní informace o objevování sousedů jsou shrnuty v RFC 4861: Neighbor Discovery for IP vision 6.

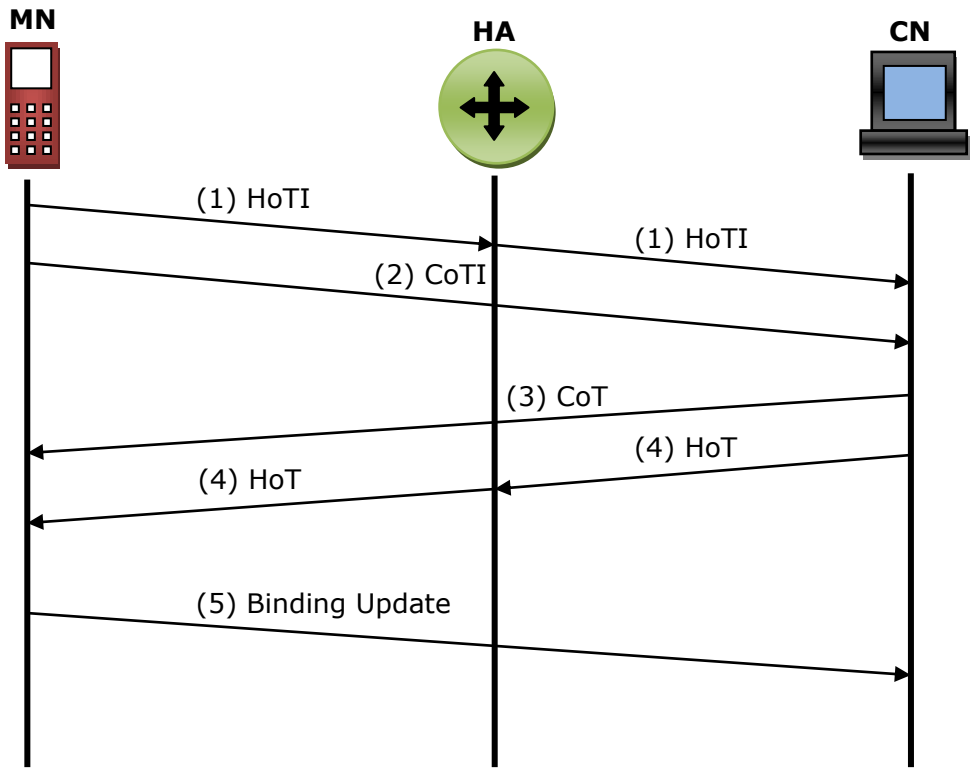
2.2.4 Objevování domácích agentů (Home Agent Discovery Mechanism)

Mobilní uživatel (MN) nemusí znát adresu svého Domácího agenta (HA) [1, 7]. Objevování HA je technika, pomocí které je MN schopen vyhledat svého HA, pokud MN nezná adresu HA nebo na současné adrese již není přístupný. Děje se tak pomocí ICMP zprávy (ICMP Home Agent Address Discovery Request Message). Tuto zprávu pošle MN na anycastovou adresu domácí sítě MN. Funkci HA může v domovské síti zajišťovat více routerů. Každý router má povinně svůj seznam, ve kterém je informace o domácích agentech (adresa, priorita) v síti. Tento seznam pošle MN prostřednictvím ICMP zprávy (ICMP Home Agent Discovery Reply Message). MN nyní bude posílat aktualizaci vazby na tyto (již unicastové) adresy. Začíná u HA s nejvyšší prioritou.

[3] Více k ARP lze nalézt na: http://cs.wikipedia.org/wiki/Address_Resolution_Protocol

2.2.5 Ověření zpětného směrování (Return Routability Procedure = RRP)

Tato procedura zabezpečuje aktualizaci vazby mezi MN a CN při optimalizovaném směrování [1, 7, 8]. Byla vyvinuta proto, aby nemohlo docházet k narušení bezpečnosti při přímé komunikaci MN-CN. RRP umožňuje CN poskytnout důkaz o tom, že MN je adresovatelný na své udávané CoA stejně dobře, jako na své domovské adrese. Pouze na základě tohoto ověření, bude CN akceptovat aktualizaci vazby mezi MN-CN. RRP probíhá dle obrázku 2.2 takto:



Obr. 2.2: Průběh RRP procedury

Proces začíná vysíláním zprávy *(1) Zahájení testu domácí adresy* (Home Test Init Message = HoTI). Zprávu HoTI vysílá mobilní uživatel přes tunel k domácímu agentovi a ke Korespondentovi. Současně se zprávou HoTI, vysílá mobilní uživatel zprávu *(2) Zahájení testu CoA* (CoTI). Zprávu CoTI vysílá mobilní uživatel přímo na adresu Korespondenta. Okamžitě po vyslání zpráv (1), (2) si mobilní uživatel uloží do Seznamu aktualizací vazeb (Binding Update List) tyto informace:

- adresu uzlu, kterému posílal zprávy HoTI resp. CoTI,
- svojí domovskou adresu,
- časy, při kterém byly zprávy odeslány,
- Cookie použité ve zprávách.

Korespondent postupně přijme obě zprávy a odpoví na ně zasláním zpráv (3) Test domácí adresy (Home Test Message = HoT) a (4) Test CoA (Care-of Adress Test Message = CoT). Zprávu HoT Korespondent zasílá na adresu domácího agenta (je uvedena v HoTI). Zprávu CoT zasílá Korespondent na adresu CoA (je uvedena v CoTI). Ve zprávě (3) resp. (4) jsou obsažena šifrovaná data soukromým klíčem Korespondenta. Tyto zašifrovaná data obsahují domácí adresu resp. CoA a vygenerovanou hodnotu „nonce". Takto zašifrovaným datům se říká „token". K tokenům je připojena informace o indexu použitého nonce a cookie[4]. Pakliže mobilní uživatel přijme zprávu HoT i CoT, může z tokenů obsažených v těchto zprávách vygenerovat hodnotu, kterou zabezpečí vazbu mezi sebou a Korespondentem. Pošle tedy takto zabezpečenou zprávu (5) Aktualizace vazby (Binding Update) na adresu Korespondenta. Korespondent autorizuje tuto zprávu pomocí uložených informací o indexech nonce, spočítá si znovu oba tokeny a jsou-li totožné s přijatými, vyhodnotí se proces Aktualizace vazby jako bezpečný. Korespondent si do své Paměti vazeb (Binding Cache) uloží adresu CoA mobilního uživatele, na kterou bude od nynější doby posílat data. Záznam v Binding Cache může vypršet po uplynutí času, definovaného v aktualizaci vazby nebo při vymazání z Binding Cache.

Procedura RRP se bude spouštět, pakliže nebude mít CN záznam o vazbě s MN. Dojde-li k přemístění MN do cizí sítě, postačuje, když MN pošle aktualizaci vazby na adresu CN.

Procedura RRP se provádí u optimalizované verze MIPv6. Pro její funkci je vyžadováno, aby síť podporovala IPv6.

Obsah zpráv definovaných v rámci RRP

Zprávy definované v rámci RRP jsou součástí hlavičky pro podporu mobility definované v rámci RFC 3775. Parametry, které tyto zprávy obsahují, jsou uvedeny v tabulce 2.1.

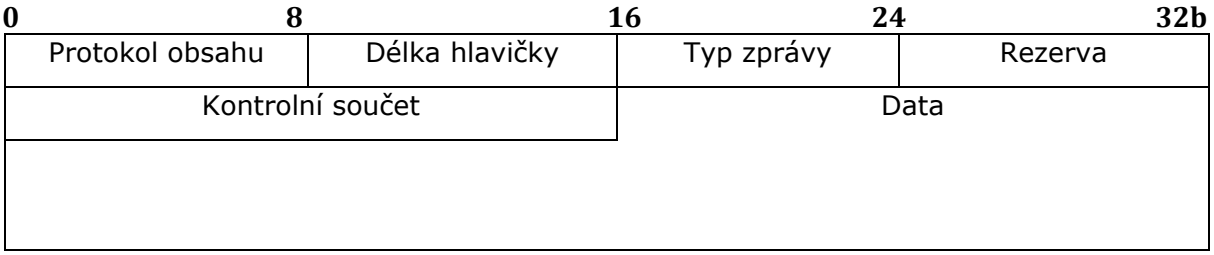
Tab. 2.1: Parametry zpráv definovaných v rámci RRP

Zpráva	**Parametry (velikost)**
HoTI	Home Init Cookie (64 bit)
CoTI	Care-of Init Cookie (64 bit)
HoT	Home Init Cookie (64 bit), Home Keygen Token (64 bit), Home Nonce Index
CoT	Care-of Init Cookie (64 bit), Care-of Keygen Token, Care-of Nonce Index

[4] Cookie je hodnota generovaná na straně mobilního uživatele a slouží k ověření autenticity MN u CN při RRP – cookie vysílaná mobilním uživatelem ve zprávě HoTI resp. CoTI musí být stejná jako přijatá ve zprávě HoT.

2.3 Zprávy a jejich hlavičky definované pro podporu mobility v IPv6

Základním stavebním prvkem je hlavička pro podporu mobility, která je definována v RFC 3775. Jedná se o rozšiřující hlavičku IPv6. Hlavička má tento tvar [1].



Obr. 2.3: Rozšiřující hlavička pro podporu mobility v IPv6

Pro nás je nejdůležitější položka Typ zprávy (MH Type). Jedná se o 8-bitový identifikátor, který určuje, jaký typ zprávy je tímto rámcem přenášen. Možnosti shrnuje následující tabulka (Tab. 2.2).

Tab. 2.2: Rozlišení jednotlivých zpráv, přenášených v základní hlavičce pro podporu mobility [8].

Typ zprávy	**Význam (originální dle RFC)**	**Význam (překlad)**
0	binding refresh request	žádost o obnovení vazby
1	home test init	zahájení testu domácí adresy
2	care-of test init	zahájení testu dočasné adresy
3	home test	test domácí adresy
4	care-of test	test dočasné adresy
5	binding update	aktualizace vazby
6	binding acknowledgement	potvrzení vazby
7	binding error	chyba vazby

Tyto zprávy používají mobilní uživatel (MN), Domácí agent (HA) a Korespondent (CN). Tabulka obsahuje všechny možnosti, definované v rámci RFC 3775. Tato tabulka se bude do budoucna rozšiřovat, jak budou definovány nové možnosti mobility. Například FMIPVv6 přidává do této tabulky ještě hodnoty 8, pro rychlou aktualizaci vazby (FBU) a 9, pro potvrzení této vazby (FBack). Podrobnosti jsou uvedeny v kapitole 3.1.

2.3.1 Žádost o obnovení vazby (binding refresh request)

Zprávu posílá CN na adresu MN [1]. MN bude reagovat na zprávu, pakliže má záznam v Binding List o adrese, z které přišla tato žádost. Žádost o obnovení vazby se pošle v případě, že má vypršet záznam v Binding Cache (na straně CN nebo HA). Pakliže by u CN vypršel záznam

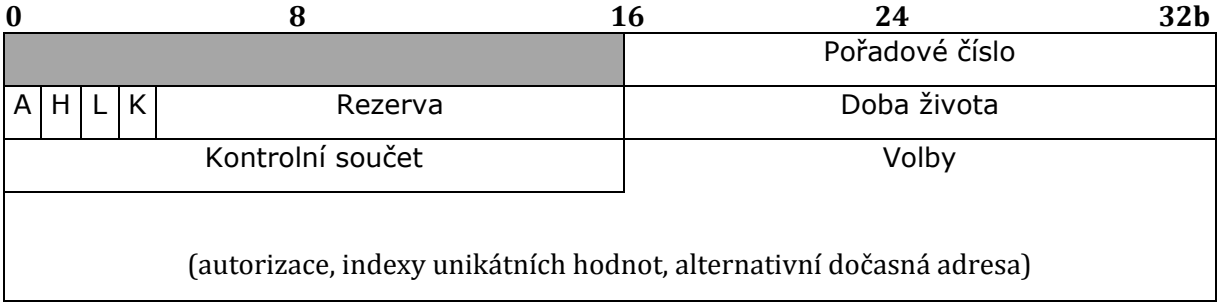
v Binding Cache dříve, než by se stihla obnovit aktualizace vazby mezi MN-CN, budou se data posílat na adresu HA a MN bude muset zahájit RRP.

2.3.2 HoTI, CoTI, HoT, CoT (zprávy 1 až 4 dle tabulky)

Jsou zprávy přenášené při RRP, viz. kapitola 2.2.5.

2.3.3 Aktualizace vazby (binding update = BU)

Aktualizace vazby se vyměňuje mezi mobilním uživatelem a jeho domácím agentem a umožňuje domácímu agentovi udržovat informace o poloze mobilního uživatele, přesněji o jeho CoA. Také ji posílá mobilní uživatel Korespondentovi, při optimalizovaném směrování MIPv6 [1].



Obr. 2.4: Hlavička zprávy BU

Pořadové číslo: Je důležité pro správné vyhodnocení žádosti o aktualizaci vazby.

A: Tento bit je nastaven při požadavku o potvrzení vazby od adresáta, kterému je BU zasílána.

H: Žádost na adresáta, aby se stal prvkem HA. Tímto se může stát pouze uzel v domácí síti.

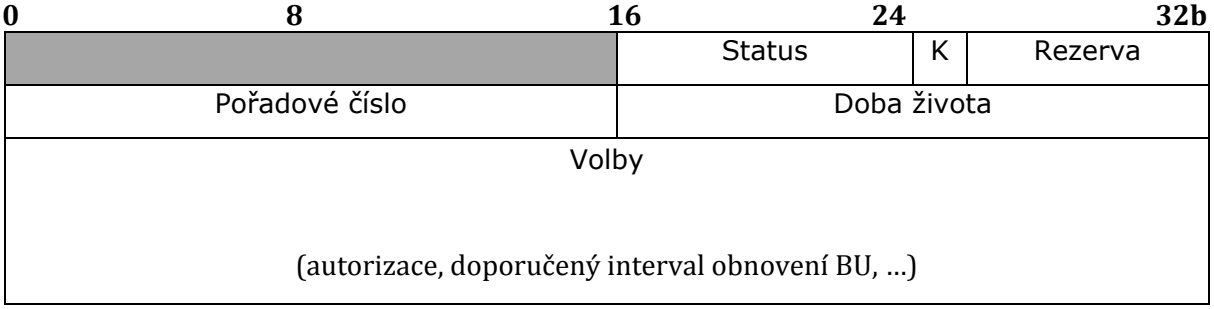
L: Tímto bitem informuje MN svého HA, že zná linkovou adresu svého HA.

K: Tímto bitem sděluje MN svému HA, že podporuje zabezpečený přenos IPSec (bližší informace o využití IPSec v MIPv6 lze nalézt v RFC 4877).

Doba života (orig. „Lifetime“)**:** Informace o tom, jak dlouho bude BU platná. Při vypršení platnosti je nutné aktualizovat vazbu. Aktualizaci vazby lze vyžádat zprávou Binding RefreshRequest.

2.3.4 Potvrzení vazby (binding acknowledgement = BAck)

Slouží pro šíření informací o výsledku aktualizace vazby [1]. Například při zasílání aktualizace vazby mobilním uživatelem na router v síti, který není nastaven pro podporu domácího agenta, vrací tento router zprávu binding acknowledgement, kde pole Status je vyplněno číslem 131 (registrace domácího agenta není podporována). Další chybové statuty jsou uvedeny v RFC 3775. Hlavička zprávy BAck je uvedena na obrázku 2.5.



Obr. 2.5: Hlavička zprávy BAck

Prostřednictvím této zprávy se také šíří informace o **době života** (orig. „Lifetime“). Tato adresátovy říká, po jakou minimální dobu bude odesílatel udržovat vazbu, než vyprší.

Ve **volbách** hlavičky se můžou šířit autorizační data (nutné v případě šíření BAck od CN) a dále doporučená doba požadovaného obnovení aktualizace. Další volby mohou přibývat.

Tato zpráva je povinná u komunikace MN-HA (je nastaven příznak H v hlavičce BU), nebo je-li nastaven příznak A v hlavičce BU, případně ve většině chyb vzniklých při aktualizaci vazby.

2.3.5 Chyba vazby (binding error)

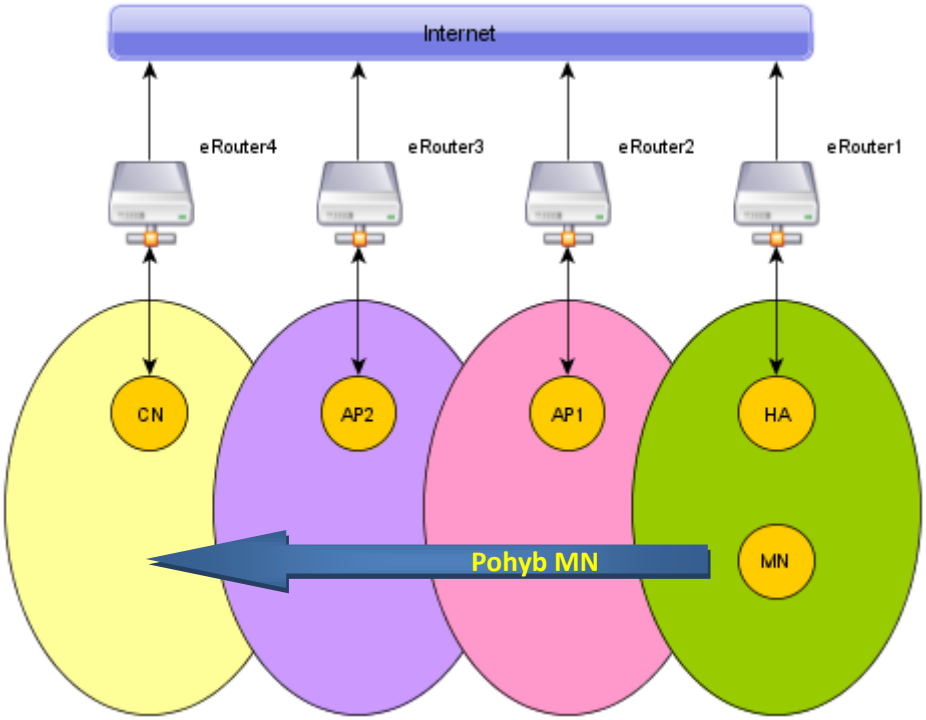
Nastane-li chyba při aktualizaci vazby, pošle CN tuto zprávu. Chybová hláška vznikne například v tomto případě: Mobilní uživatel se snaží aktualizovat vazbu s Korespondentem, ale výsledek RRP je nežádoucí [8].

2.4 Topologie MIPv6

Mobilní síť v IPv6 (základní) se skládá z mobilního uživatele (MN), domácího agenta (HA) a korespondenta (CN). Obrázek 2.6 znázorňuje topologii MIPv6.

2.4.1 Korespondent (CN)

Korespondent vystupuje ve funkci odesílatele dat k mobilnímu uživateli. Při odesílání dat nahlédne do záznamu v Binding Cache, pakliže je zde odpovídající záznam o CoA mobilního uživatele, pošle data na adresu CoA. Využije k tomu směrovací hlavičku typu 2. Pakliže v Binding Cache není odpovídající záznam o CoA mobilního uživatele, pošle data standardně (nepoužije se směrovací hlavička typu 2) na adresu domácího agenta. CN neví, že se chystá komunikovat s mobilním uživatelem [1]. Korespondent se dále stará o ověřování zprávy Aktualizace vazby mezi MN-CN. Proces je popsán v 2.2.5. CN také spravuje záznamy v paměti vazeb (Binding Cache).



Obr. 2.6: Topologie MIPv6

2.4.2 Domácí agent (HA)

Zastupuje při komunikaci mobilního uživatele. Komunikace mezi MN-HA je zabezpečena pomocí IPsec. Stejně jako CN si udržuje seznam Bindign Cache, kde ukládá informace o CoA mobilního uživatele a navíc si udržuje ještě Seznam domácích agentů (Home Agent List). Seznam domácích agentů slouží pro udržení informací o domácích agentech v síti a je používán při technice objevování domácích agentů, vysvětleno v kapitole 2.2.4.

2.4.3 Mobilní uživatel (MN)

Je uživatel, který se buď pohybuje v domácí síti a je standardně adresovatelný nebo je mimo svojí domovskou síť a je zastupován domácím agentem. MN umí vyhledat svého HA pomocí mechanizmu objevování domácích agentů, popsáno v 2.2.4, umí také reagovat na změny adres v domovské síti. MN se stará o aktualizaci vazby mezi sebou a HA. MN také zahajuje proceduru RRP. Komunikace mezi MN-HA je zabezpečena pomocí IPsec (ESP hlavička, případně ESP+AH) [1, 7, 8].

2.4.4 Popis pohybu MN v topologii MIPv6

MN je v domácí síti

Pakliže je mobilní uživatel v domácí síti, probíhá provoz standardně. Korespondent bude komunikovat přímo s mobilním uživatelem, protože ten bude dostupný na svojí domácí adrese.

MN vycestuje do cizí sítě

MN získá adresu cizí sítě, děje se standardními postupy, definovanými v IPv6 (autokonfigurace stanice, zapojí se také zprávy router advertisement, solicitation), najde svého domácího agenta pomocí objevování sousedů nebo má jeho adresu staticky definovanou. Informuje o této dočasné adrese (CoA) domácího agenta pomocí aktualizace vazby (BU). MN Je postupně připojován na jednotlivé přístupové body (AP) cizích sítí.

MN se vrací do domácí sítě

MN zruší vazbu na domácího agenta a opět přijímá data na své domácí adrese.

2.5 Komunikace mezi korespondentem a mobilním uživatelem (bez optimalizace)

Základní myšlenka MIPv6 je stejná jako u MIPv4, tedy aby mobilní zařízení zastupoval směrovač v domácí síti, pakliže mobilní uživatel vycestuje do cizí sítě [12]. Při navazování komunikace pak tento směrovač zachytává provoz určený pro mobilní zařízení a přepošle ho tunelem k mobilnímu zařízení [8]. Pro tento směrovač v domácí síti se používá označení Domácí agent. Uživatel, který chce komunikovat s mobilním uživatelem, se nazývá Korespondent. Domácí agent udržuje informaci o poloze mobilního uživatele. Rozdíl oproti MIPv4 je ten, že FA v cizí síti již není potřeba [12]. Obrázek 2.7 popisuje zprávovou výměnu v MIPv6 (bez optimalizace).

2.5.1 Popis komunikace

Registrace MN u HA

MN se zaregistruje u svého domácího agenta (HA). Pakliže nezná adresu svého HA, může ho vyhledat pomocí zprávy ICMP Home Agent Address Discovery Request Message (jedná se o metodu objevování domácích agentů). Získanou adresu pak zaregistruje pomocí Binding Update. Zprávy Binding Update a Binding Acknowledgement se mezi HA a MN přenáší bezpečným tunelováním (ESP nebo ESP+AH hlavička). [1]

MN je doma

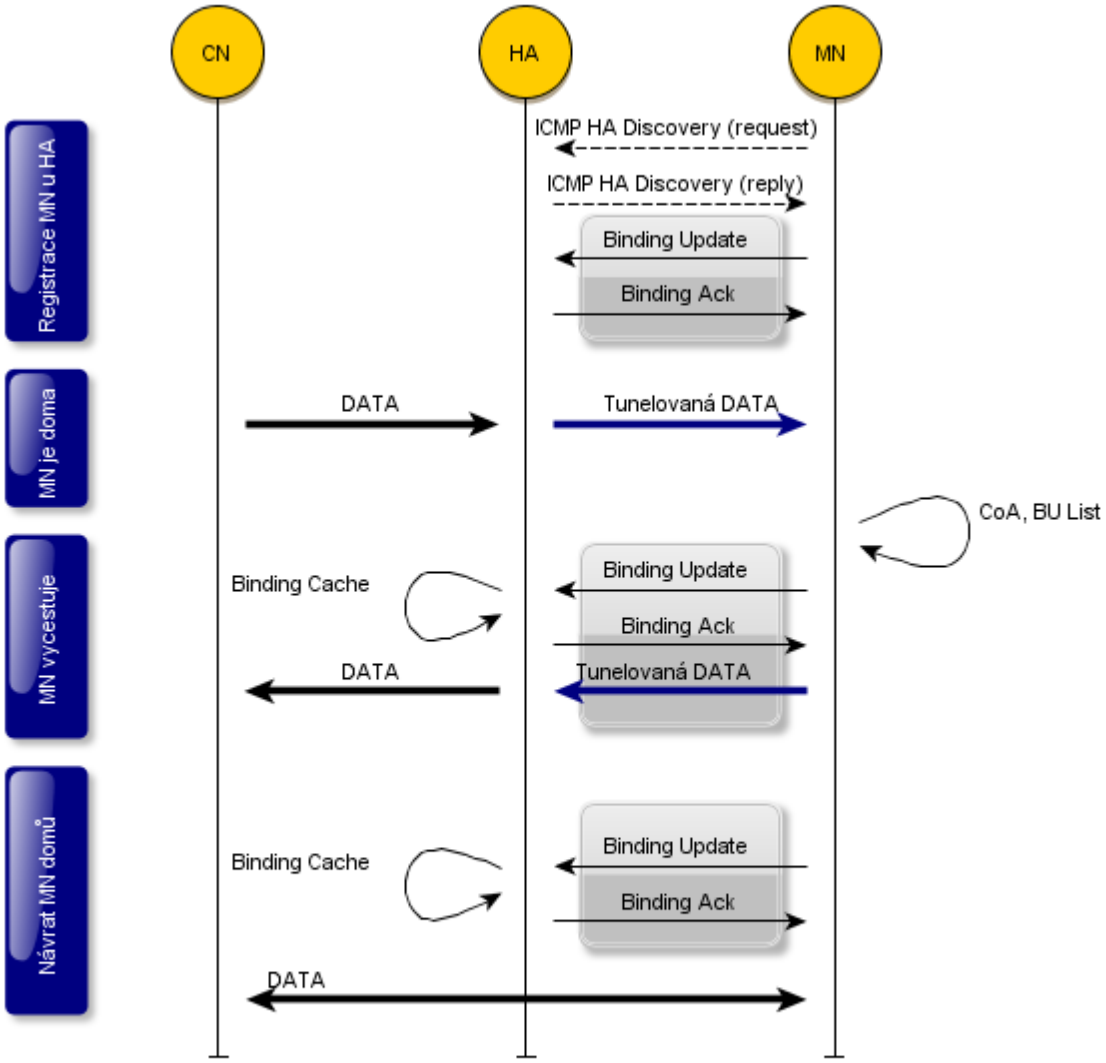
Uživatel (CN), který chce komunikovat s MN, je nejprve spojen s HA. HA je umístěn v domovské síti MN. Pakliže je MN doma, probíhá komunikace standardně.

MN vycestuje

Pakliže je MN mimo domovskou síť, získá novou CoA a tuto adresu musí oznámit svému HA. Informace o proběhlých aktualizacích vazby se na straně MN ukládají v Binding Update List. Na straně HA existuje Binding Cache, kde se také uloží informace o aktualizaci vazby. HA pošle potvrzení (BAck) a od této chvíle budou data od CN odesílána k MN přes tunel mezi HA-MN. Toto přeposílání dat je transparentní, CN neví, že komunikuje s Domácím agentem a ne s Mobilním uživatelem. Naopak, veškerá komunikace ze strany Mobilního uživatele, je tunelována do HA a dále pak běžně šířena k CN [1, 8].

Návrat MN domů

Při návratu domů pošle MN aktualizaci vazby (BU), aby vymazal záznam u HA. HA bude od této chvíle posílat data k MN standardně, tunel již není potřeba. V hlavičce BU je nastaven bit H, A a CoA adresa ukazuje na domovskou adresu a doba života je nastavena na 0. Po zrušení tunelu je nutné, aby MN poslal ohlášení souseda a nebyly tak data posílány na linkovou adresu HA [1, 8].



Obr. 2.7: Zprávová výměna při komunikaci CN-MN v základním MIPv6

2.6 Komunikace mezi korespondentem a mobilním uživatelem (s optimalizací)

Optimalizace spočívá v informování Korespondenta o aktuální adrese (CoA) mobilního uživatele. Oproti standardní MIPv6 přibyla procedura RRP (je popsána v 2.2.5).

2.6.1 Popis komunikace

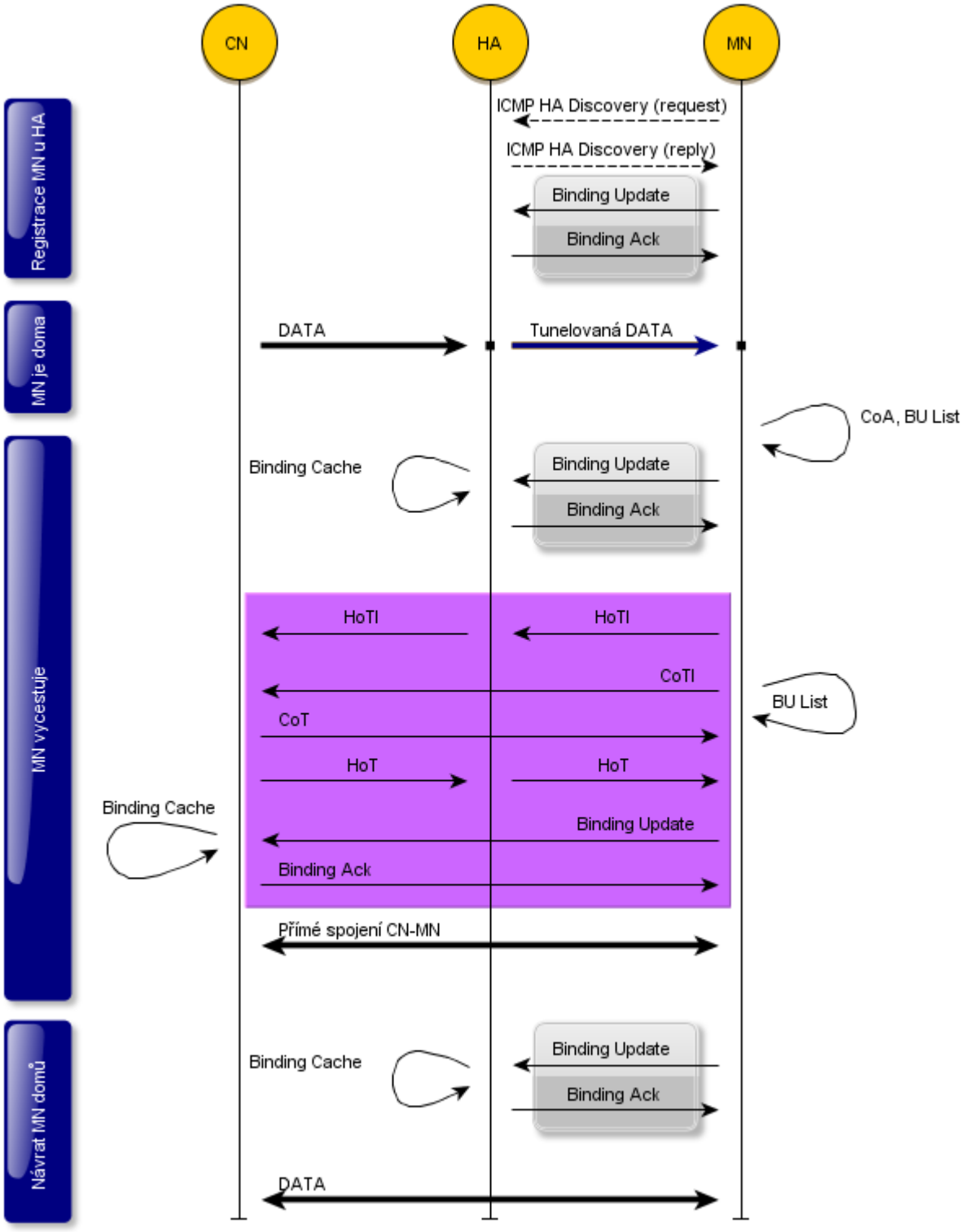
Následující popis se vztahuje k obrázku 2.8.

Proces **registrace MN u HA** je totožný jako u standardní MIPv6 v kapitole 2.5.

MN v domácí síti přijímá data stejně jako u standardní verze MIPv6 (popsáno v kapitole 2.5).

Pakliže **MN vycestuje mimo domácí síť**, musí u optimalizované verze MIPv6 informovat kromě svého HA také uživatele CN. Po přijetí potvrzení vazby od HA je (v případě, že se MN ještě neregistroval u CN) zahájena procedura RRP (v obrázku 2.8 proceduru RRP znázorňuje fialová oblast). Tato procedura je popsána v kapitole 2.2.5. Při přecházení MN není nutné znovu spouštět proceduru RRP, pakliže informuje CN o nové CoA. Aktualizace vazby musí vždy proběhnout mezi MN-HA [1]. Po registraci MN u CN bude komunikace probíhat přímo mezi CN a MN [1].

Návrat domů pro MN znamená, že pošle BU svému HA, ten zruší tunel a od této chvíle bude posílat data pro MN přímo [1]. V hlavičce BU je nastaven bit H, A, CoA adresa ukazuje na domovskou adresu a doba života je nastavena na 0. Po zrušení tunelu je nutné, aby MN poslal ohlášení souseda a nebyly tak data posílány na linkovou adresu HA [1, 8].



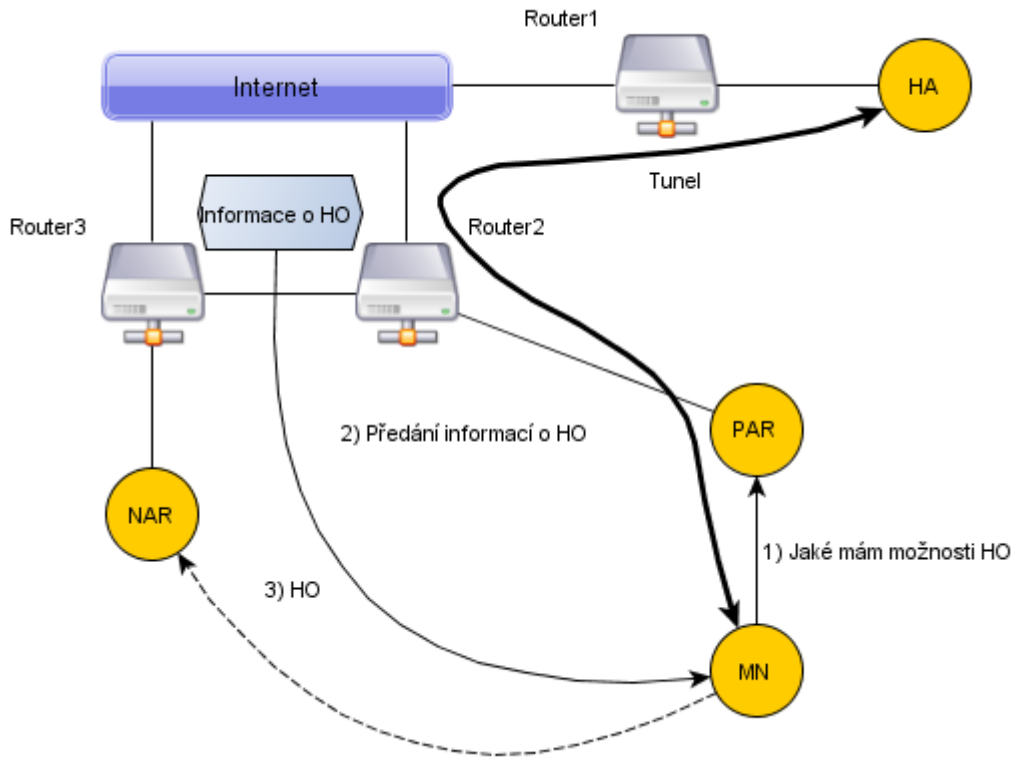
Obr. 2.8: Zprávová výměna při komunikaci CN-MN v optimalizovaném MIPv6

2.7 Shrnutí k MIPv6

Mobilita v rámci IPv6, tak jak je definována v RFC 3775, nevyhovuje pro nasazení některých služeb. Vstoupí-li mobilní uživatel do cizí sítě, musí získat a zaregistrovat dočasnou (CoA) adresu (pomocí autokonfigurace stanice). Tyto procedury trvají určitou dobu a nelze je zanedbat v okamžiku, kdy požadujeme přenos dat v reálném čase, tedy například přenos hlasu/videa [8, 13]. Vznikly další metody, které se liší v řešení handoveru, a také z topologického hlediska. V následujícím textu bude řeč o Hierarchické mobilitě (HMIPv6), a také poskytování rychlého handoveru (tzv. „fast handover“) v MIP i HMIP. Tyto metody se snaží vylepšit MIPv6.

3 RYCHLÝ HANDOVER V MIPV6 [FMIPV6]

Podporou rychlého handoveru se zabývá specifikace RFC 5268 z června 2008. Technika rychlého handoveru spočívá v získání nové CoA (tzv. NCoA), dříve než dojde ke ztrátě současné CoA. Nová CoA bude dočasná adresa sítě, do které bude mobilní uživatel vstupovat. Tímto je snížena doba odezvy při vyřizování nového přihlášení k cizí síti, když mobilní uživatel opouští starou síť [2]. Schématické znázornění topologie FMIPv6 je na obrázku 3.1.



Obr. 3.1: Topologie FMIPVv6 – počátek pohybu MN

V souvislosti s technikou rychlého handoveru jsou definovány tyto pojmy [2]:

- **Přístupový router** (orig. „Access Router" (AR)) je posledním prvkem v síti, který tvoří spojení mezi mobilním uživatelem a sítí. MN je na linkové vrstvě spojen s AR.
- **Předchozí přístupový router** (orig. „Previous Access Router" (PAR)) je při pohybu mezi starým a novým přístupovým routerem považován za výchozí.
- **Nový přístupový router** (orig. „New Access Router" (*NAR*)) je router, na který bude MN provádět handover.
- **Předchozí dočasná adresa** (orig. „Previous Care-of Adress" (*PCoA*)) je adresa, kterou MN vlastní při pobytu v síti PAR do okamžiku, než opustí tuto síť a získá NCoA.
- **Nová dočasná adresa** (orig. „New Care-of Adress" (*NCoA*)) je adresa, kterou MN získá při vstupu do sítě NAR.
- **Handover** je pojmenování pro proces přepojení MN z jednoho routeru na druhý. V metodě FMIPVv6 je MN přepojován mezi PAR a NAR.

FMIPVv6 má dva funkční režimy. Prediktivní a reaktivní. Tyto režimy se liší v tom, že v prvním režimu, MN stihne všechnu potřebnou zprávovou výměnu potřebnou pro přepojení na NAR. V druhém režimu, MN posílá zprávu FBU až při vstupu do sítě NAR, protože jeho přesun je příliš rychlý. Procesy komunikace v prediktivním a reaktivním režimu jsou popsány v kapitole 3.2, resp. 3.3.

3.1 Zprávy a jejich hlavičky definované v rámci FMIPv6

Pro techniku FMIPVv6 jsou definovány nové zprávy, které nejsou známy v RFC 3775. Zprávy jsou v přehledu níže. Bližší informace jsou k nalezení v RFC 5268 [2].

RtSolPr

- Kam: Posílá ji MN multicastově nebo na adresu PARu.
- Účel: MN potřebuje znát informace o síti, na kterou bude provádět HO. Touto zprávou si informace vyžádá od PARu, které má tyto informace k dispozici.

PrRtAdv

- Kam: Posílá ji PAR na adresu MN.
- Účel: Poskytnutí informací o možnostech HO a možnosti zformovat NCoA.

FBU

Definována stejně jako zpráva aktualizace vazby (Binding Update) v RFC 3775. Pravidla, kterým podléhá, při jejím zpracování jsou mírně odlišná.

HI

- Kam: Posílá ji PAR na IP adresu NARu.
- Účel: Inicializace handoveru. PAR si může touto zprávou požádat o NCoA pro MN nebo nemusí, zjistí-li MN adresu NCoA ze zprávy PrRtAdv. Použije se v případě prediktivního režimu FMIPv6. Může si také požádat o ukládání paměti do bufferu.

HAck

- Kam: Posílá ji NAR na IP adresu PARu.
- Účel: Reakce na HI. Přenáší výsledek o přijetí/zamýtnutí NCoA. Touto zprávou také může NAR NCoA doporučit. Pakliže je přijata HI, musí být odeslána HAck.

FBack

Definována stejně jako zpráva potvrzení vazby (Binding Acknowledgement) v RFC 3775. Pravidla, kterým podléhá, při jejím zpracování, jsou mírně odlišná.

- Kam: Posílá se z adresy doručení FBU na NCoA a volitelně i na PCoA.
- Účel: Potvrzení FBU.

UNA

Definována v rámci RFC 4861. V RFC 5268 mírně upravena.

- Kam: Přenášena z adresy NCoA na IP adresu NARu nebo multicastově.
- Účel: Oznámení sousedům, že do sítě přišel MN.

Tabulka 3.1 stručně popisuje obsah těchto zpráv.

Tab. 3.1: Důležité parametry hlaviček

RtSolPr	Ve volbách (orig. „options“) může obsahovat linkovou adresu odesílatele (MN) a musí obsahovat linkovou adresu příjemce (PARu).
PrRtAdv	Kopíruje některé adresy z RtSolPR. Odesílá ve volbách AP-ID a AR-INFO. Obsahuje položku Code, která rozhoduje, jak se zprávou bude dále nakládáno. Code = 0: PrRtAdv obsahuje AP-ID a AR-INFO. Code = 1: Zpráva zaslána nevyžádaně ➔ MN musí použít NCoA, doručenou v této zprávě a okamžitě poslat FBU, jinak ztrácí možnost připojit se k AP, které PrRtAdv vyslal. Jsou ještě další možnosti Code. [2]
FBU	V hlavičce pro podporu mobility, má přiřazeno číslo 8 v parametru Type. Rozšiřuje tak Tab. 2.2. Bit „H“ musí mít nastaven na jedna[5]. Ve volbách musí být nastavena alternativní CoA na NCoA, pakliže je FBU vyslána z domény PARu a musí obsahovat autorizační data.
HI	Bit „S“ umožňuje vyžádat si v odpovědi na tuto zprávu adresu NCoA. V případě Code = 0 ve zprávě HI, může být NCoA poslána, když Code = 1, nesmí být NCoA poslána. Ve volbách hlavičky je možné přenášet NCoA vygenerovanou uživatelem MN. Ve volbách hlavičky se také musí přenášet informace o MAC adrese MN a měla by se přenášet IP adresa PARu. Bit „U“ rozhoduje o ukládání přijatých paketů (od chvíle, kdy HI dorazí) na straně NARu.

[5] Může se dokonce stát, že NAR pošle IP adresu prvku PAR. V tomto případě bude NAR vyloučen z budoucí komunikace a MN na něj ani nebude posílat FBU.

HAck	Ve volbách může přenášet NCoA doporučovanou pro MN.
FBack	V hlavičce pro podporu mobility, má přiřazeno číslo 9 v parametru Type. Rozšiřuje tak Tab. 2.2. Volba také obsahuje autorizační data.
UNA	Bit „O“ bude v FMIPVv6 vždy nastaven na 0.

3.2 Popis procesu komunikace prediktivního režimu FMIPVv6

3.2.1 Popis komunikace

Popis níže uvedený se vztahuje k obrázku 3.2.

Standardní procedury MIPv6

V obrázku jsou zeleně vyznačeny procedury, které jsou definovány v rámci MIPv6. Tyto procedury proběhnou před samotným spuštěním rychlého handoveru.

Nejprve proběhne registrace MN u HA, prostřednictvím zpráv BU a Back. Poté MN vycestuje mimo domovskou síť.

Registrace v PARu

Nachází se nyní v síti PARu a potřebuje zde získat PCoA. PCoA získá standardními technikami definovanými v IPv6 (autokonfigurací) a to buď pomocí zprávy Router Advertisement, kterou v pravidelných intervalech vysílá PAR nebo pomocí zprávy Router Solicitation, pomocí které si vyžádá informace o síti, ve které se právě nachází.

MN získal PCoA a tuto zaregistruje u svého HA postupem, definovaným v rámci MIPv6. Je-li požádován optimalizovaný přenos, je nutné, aby MN zaregistroval svojí PCoA také u CN. Zde se uplatní opět postup řešený v rámci MIPv6.

Informace o dosažitelných handoverech.

Přístup FMIPVv6 je v tomto směru odlišný od MIPv6. Možností, jak získat informaci o možnostech HO pro MN je více. Především se může jednat o HO, který je inicializován sítí nebo je inicializován přímo uživatelem MN [RFC FMIPv6]. Bohužel specifikace FMIPVv6 neřeší problematiku, jak MN zjistí informaci o dostupnosti nového AP před rozpadem spojení. Řešením je použití „triggeru“ (spouštěče) z druhé vrstvy (linkové) ISO-OSI [19].

Komunikace začíná od MN, který posílá zprávu RtSolPr svému nadřazenému routeru v cizí síti – prvku PAR [2]. Touto zprávou si MN žádá o poskytnutí informací o možnostech handoveru do cizí sítě, začne hledat potencionálního následníka, prvek NAR[6]. PAR odpoví zprávou PrRtAdv, ve které je obsažena informace o dostupných (jednom nebo více) routerech,

[6] Ve specifikaci RFC 5268 není striktně vyžadováno, aby komunikace začala zprávou RtSolPR. PAR může periodicky vysílat zprávy PrRtAdv a pakliže se tato zpráva dostane k MN dříve, než on vyšle zprávu RtSolPr, započne komunikace F-MIPv6 zprávou PrRtAdv.

které mohou plnit funkci prvku NAR. Tato zpráva obsahuje AP-ID, zastoupena v bezdrátové síti identifikátorem BSSID přístupového bodu a dále AR-info, která obsahuje linkovou adresu routeru, IP adresu a prefix sítě.

Do doby, než bude mít MN odsouhlasenou novou adresu CoA (NCoa) je přístupný na PCoA, která leží v doméně PAR a kterou znají prvky HA a CN [2].

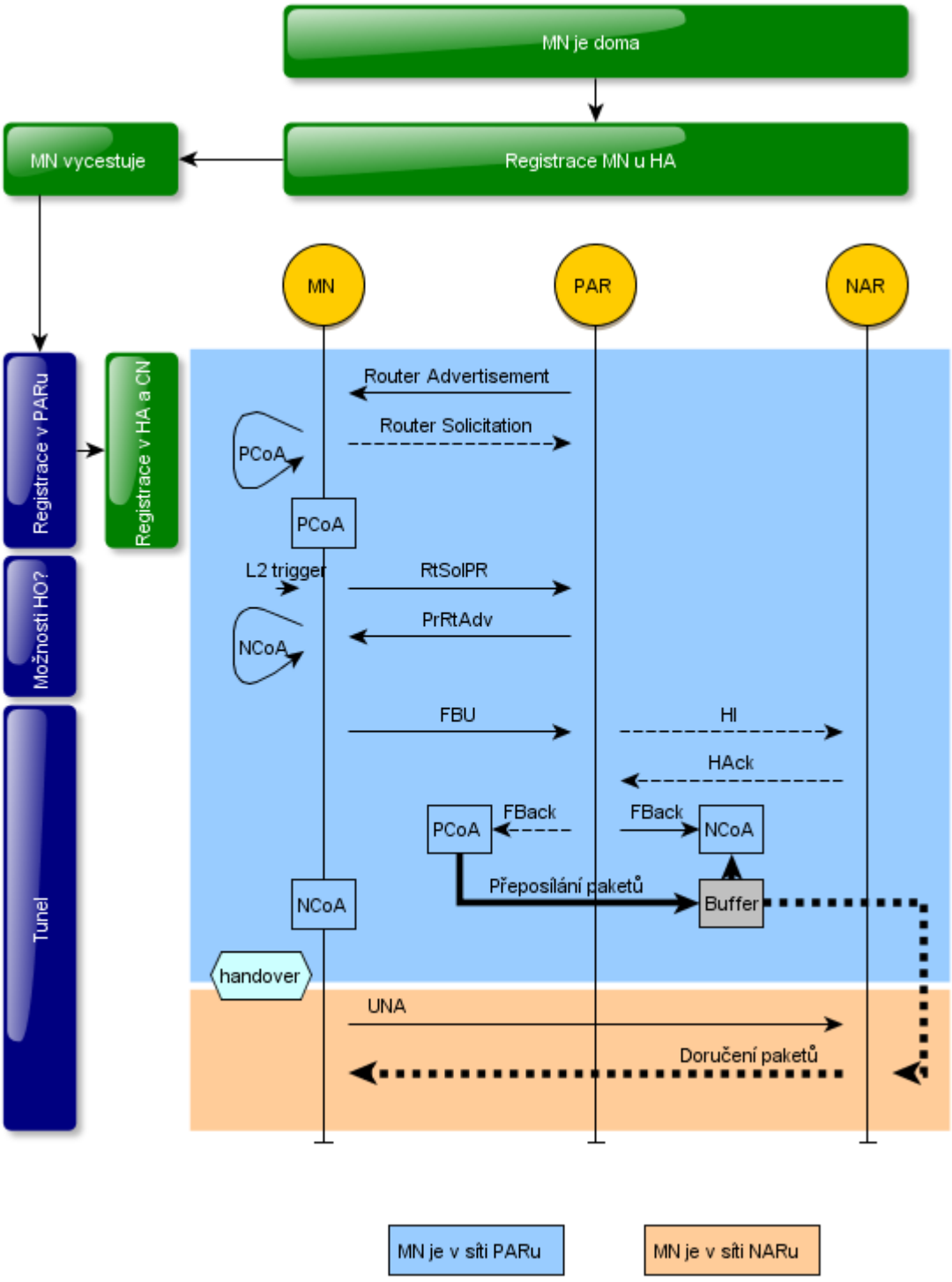
Vytvoření tunelu mezi PCoA a NCoA

Ze zprávy PrRtAdv si MN vybere vhodný router, který by chtěl mít jako NAR a na základě informací přenášených v této zprávě si vygeneruje vlastní NCoA. MN požádá o registraci této NCoA pomocí zprávy FBU. MN pošle zprávu FBU na adresu routeru PAR. Účelem FBU je, aby PAR vytvořil tunel mezi PCoA a NCoA adresou, tedy aby MN mohl být dostupný na nové adrese. A opačně, aby pakety posílány od MN s adresou NCoA, nebyly routerem PAR ignorovány a byly odesílány s adresou PCoA ven ze sítě, směrem k CN. V prediktivním režimu FMIPVv6 je zpráva FBU bezchybně odeslána z domény PAR. [2]

PAR může poslat zprávu HI, pomocí které ověří u routeru NAR možnost použití NCoA ve své síti. Odpovědí na tuto zprávu je HAcK odeslaná z NAR do PAR.

PAR pak odešle potvrzení FBU (FBack) na NCoA, volitelně také na PCoA. Po přijetí potvrzení FBack je vytvořen tunel mezi PCoA a NCoA. Na straně NARu jsou pakety z adresy PCoA ukládány do bufferu a až se MN přihlásí do sítě NARu, budou mu pakety doručeny.

Posledním krokem je ohlášení MN v nové síti. MN se ohlásí routeru NAR pomocí zprávy UNA. Jestliže je schválena jeho NCoA, bude registrace úspěšná. Pakliže se použije HI a HAck zpráva, bude platná NCoA známá, ještě před posíláním UNA. Po úspěšné registraci NCoA u NARu jsou doručena data, která byla přeposílána tunelem z PCoA na NCoA.



Obr. 3.2: Zprávová výměna při komunikaci MN s prvky PAR a NAR v rámci prediktivního FMIPVv6

3.3 Popis procesu komunikace reaktivního režimu FMIPVv6

3.3.1 Popis komunikace

Popis níže uvedený se vztahuje k obrázku 3.3.

Standardní procedury MIPv6

Stejné jako v kapitole 3.2.1.

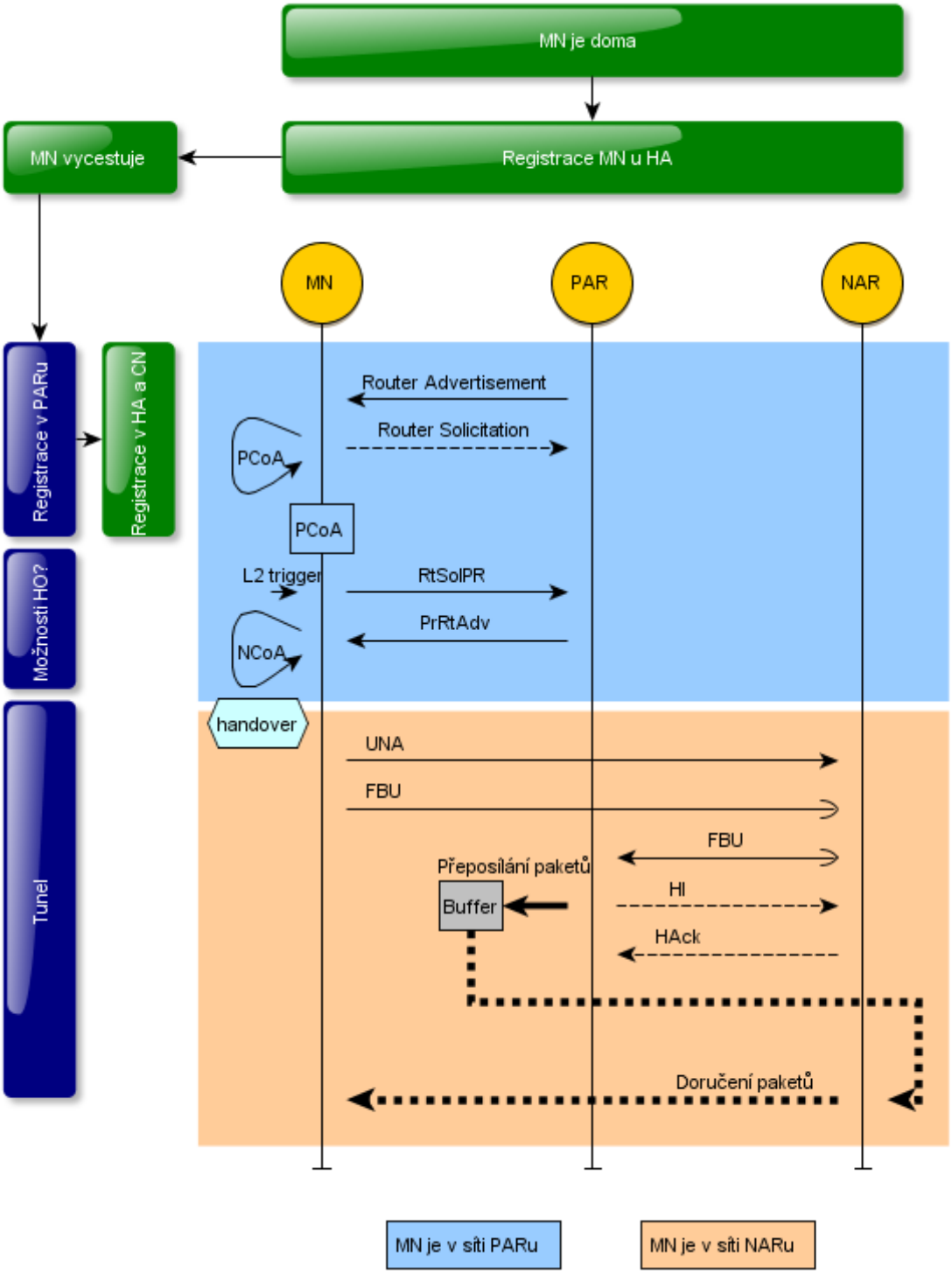
Informace o dosažitelných handoverech

Začíná stejně jako u prediktivního režimu. MN pošle RtSolPR, žádá o informace o okolních NAR. PAR poskytne tyto informace v odpovědi PrRtAdv (více v kap. 3.2.1).

Vytvoření tunelu mezi PCoA a NCoA

Ze zprávy PrRtAdv si MN vybere vhodný NAR a vytvoří si pro něj NCoA. Odtud je však komunikace odlišná od prediktivního režimu. MN se přesunul rychle a už není v dosažitelnosti prvku PAR. V reaktivním režimu FMIPVv6 je zpráva FBU odeslána na adresu PAR ze sítě NAR. Posloupnost zpráv je taková, že MN při vstupu do sítě NAR odešle UNA, kterou se ohlásí v nové síti. Pomocí zprávy FBU, kterou odešle hned po odeslání UNA, řekne prvku NAR, že chce přijímat pakety z prvku PAR na nově vytvořené NCoA adrese. Pokud by NAR zamítl vygenerovanou adresu NCoA uživatelem MN, může NAR poslat „router advertisement" s volbou NAACK obsahující novou IP pro MN. Tuto adresu pak MN musí použít jako novou NCoA.

Po odeslání zprávy UNA následuje FBU, která je posílána přes NAR, na adresu PAR. Účelem FBU je, aby PAR vytvořil tunel mezi PCoA a NCoA adresou, tedy aby MN mohl být dostupný na nové adrese. A opačně, aby pakety posílány od MN s adresou NCoA, nebyly routerem PAR ignorovány a byly odesílány s adresou PCoA ven ze sítě, směrem k CN. PAR pak pakety, které zachytával do bufferu, přepošle na tuto novou adresu, do NARu. Je-li to nezbytné, může PAR poslat zprávu HI, odpovědí bude HAck. Následuje přeposílání paktů z PAR (PCoA) do NAR (NCoA). [2]



Obr. 3.3: Zprávová výměna při komunikaci MN s prvky PAR a NAR v rámci reaktivního FMIPVv6

3.4 Shrnutí k FMIPVv6

Výsledkem je tedy to, že provoz náležící MN je z vnějšku cizí sítě přeposílán přes PAR do NAR a naopak, že provoz náležící CN je posílán z MN z NAR do PAR a dále pod výchozí PCoA k CN. Rozšířením oproti MIPv6 je zde podpora rychlého handoveru mezi dvěma routery, při cestování MN. Prediktivní režim se zdá být výhodnější, protože parametry budoucího spojení v síti NAR jsou již známy v době, kdy MN ještě komunikuje s prvkem PAR. Reaktivní režim nastává tehdy, nestihne-li MN poslat FBU ze sítě PAR. Pakliže je provedena celá popsaná technika rychlého handoveru, může dále probíhat komunikace přímo mezi CN-MN. Postačuje, aby MN poslal aktualizaci vazby na CN. Proběhne registrace MN u CN (RRP procedura) případně dojde pouze k aktualizaci vazby, pakliže spolu CN a MN již komunikovali. FMIPv6 je tak nástavbou na MIPv6, která je s MIPv6 kompatibilní.

4 HIERARCHICKÁ MOBILITA [HMIPv6]

Specifikace řešící hierarchii mobility v IPv6 se jmenuje RFC 5380. Mobilitu v IPv6 rozšiřuje o možnost připojení mobilního účastníka na více bodů v cizí síti. Hierarchická mobilita v IPv6 zavádí tyto nové pojmy [3, 8]:

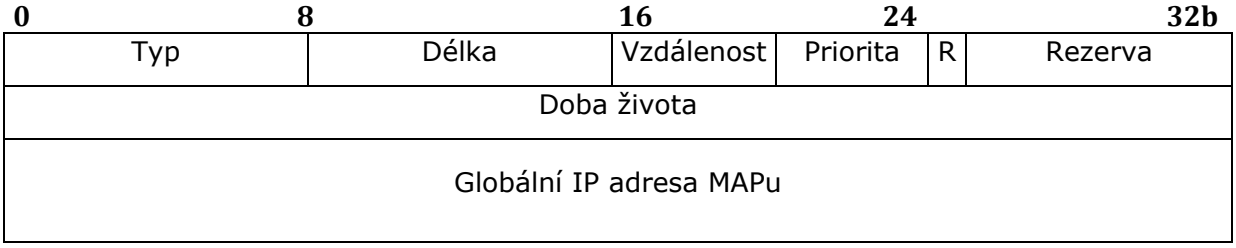
- **Kotevní bod** [orig. „Mobility Anchor Point“ (*MAP*)], je pojmenování pro jeden nebo více routerů v cizí síti, které plní funkci přípojného místa pro mobilního uživatele vstupujícího do této sítě. MAP de facto plní funkci domácího agenta v této cizí síti.
- **Regionální dočasná adresa** [orig. „Regional Care-of Address“ (*RCoA*)] je adresa kotevního bodu, na které je nepřímo dostupný mobilní uživatel.
- **Linková dočasná adresa** [orig. „On-Link Care-of Address“ (*LCoA*) je konkrétní adresa mobilního uživatele v cizí síti. Tato adresa je zaregistrována v kotevním bodu.
- **Lokální aktualizace vazby** [orig. „Local Binding Update“ (*LBU*)] je zpráva, která je zasílána mobilním uživatelem kotevnímu bodu, za účelem aktualizace své aktuální dostupnosti. Mobilní uživatel touto zprávou kotevnímu bodu sděluje, na jaké LCoA je dostupný.

4.1 Důležité hlavičky definované pro HMIPv6

4.1.1 Objevení MAPu

MN si z ohlášení směrovače vybere nejvhodnější MAP, ke kterému se připojí a získá u něj RCoA. Hlavička pro objevení MAPu je definována jako rozšířená volba hlavičky objevování sousedů.

Vypadá takto:



Obr. 4.1: Hlavička zprávy Objevení MAPu.

Významy jednotlivých položek hlavičky jsou:

Typ: 23

Délka: 3

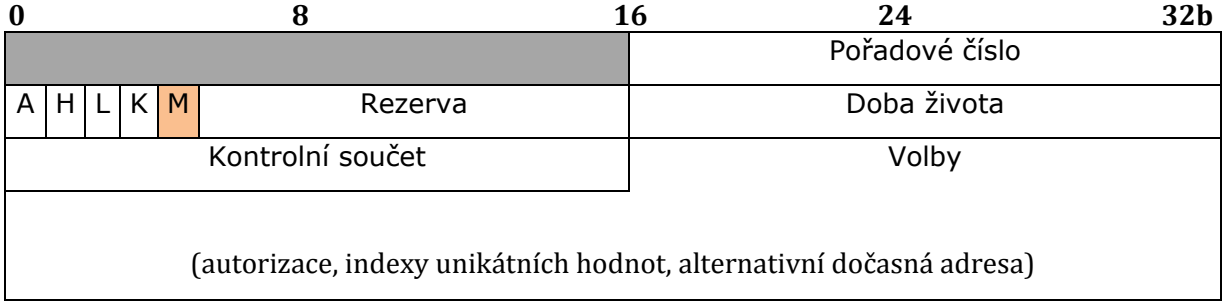
Vzdálenost: Udává jak daleko je MN od MAPu, u kterého by mohl žádat registraci RCoA. Vzdálenost je dána počtem přeskoků mezi prvky v síti mezi prvky MAP a MN. Nesmí být použita vzdálenost = 0. Může být využita při výběru vhodného MAPu.

Priorita: Slouží k usnadnění výběru nejvýhodnějšího MAPu v síti.

R: MAP se identifikuje uživateli MN, jako jím používaný. V souvislosti s tímto je možné řídit, jak se bude MN chovat při registraci RCoA u nových MAPů. Jsou definovány metody „Eager" (dychtivý) a „Lazy" (líný). V prvním režimu, provede handover na nový MAP, jakmile zjistí, že je pro něj výhodnější, než stávající MAP (toto zjistí na základě ohlašování sousedů, které se k němu dostávají pravidelně od různých routerů v síti). V druhém režimu, se MN připojí k novému MAPu, až když ztratí spojení se stávajícím (podrobnosti v RFC 5380).

4.1.2 Formování adresy RCoA a LCoA

Jakmile MN vstoupí do nové MAP domény, musí nakonfigurovat dvě adresy. Jsou to adresy RCoA a LCoA. Obě adresy vznikají na základě autokonfigurace stanice (definováno v rámci IPv6). V souvislosti s tímto rozšiřuje RFC 5380 hlavičku pro podporu mobility, která vypadá nyní takto:



Obr. 4.2: Hlavička pro podporu mobility (upravená pro HMIPv6).

Přibyl zde bit M, kterým je ve zprávě BU oznamováno, že se jedná o registraci u MAP. LCoA adresa je oznamována MN prvku MAP na adresu RCoA. Kromě M bitu je nastaven také A bit, tudíž je od MAPu vyžadováno potvrzení LBU zprávou Back.

4.2 Topologie HMIPv6

Jak je řešena komunikace mezi CN-HA-MN, je vidět na obrázku 4.3. CN bude komunikovat s MN přes hierarchickou topologii.

Je zaveden nový prvek v topologii, tzv. kotevní bod („mobility anchor point“) [8]. Může existovat více těchto kotevních bodů spadajících do jedné sítě (na obrázku níže spadají pod MAP1 kotevní body MAP2 a MAP3). Při pohybu mobilního účastníka v takovéto síti, se bude připojovat na jednotlivé kotevní body, bez nutnosti měnit parametry u svého domácího agenta. Pouze změní informace o dostupnosti u kotevního bodu. Tímto je dosaženo snížení přenášené signalizace v síti směrem k HA a CN.

Důležitá změna oproti MIPv6 je ta, že provoz k MN je tunelován dvakrát. Poprvé standardně od HA do MAP1 na regionální adresu RCoA, podruhé je tunelován od MAP1 do MN na linkovou adresu LCoA [3, 8].

4.3 Popis procesu komunikace v HMIPv6

Budeme-li se držet topologie dle obrázku 4.3, bude komunikace probíhat dle obrázku 4.4 takto:

Získání LCoA

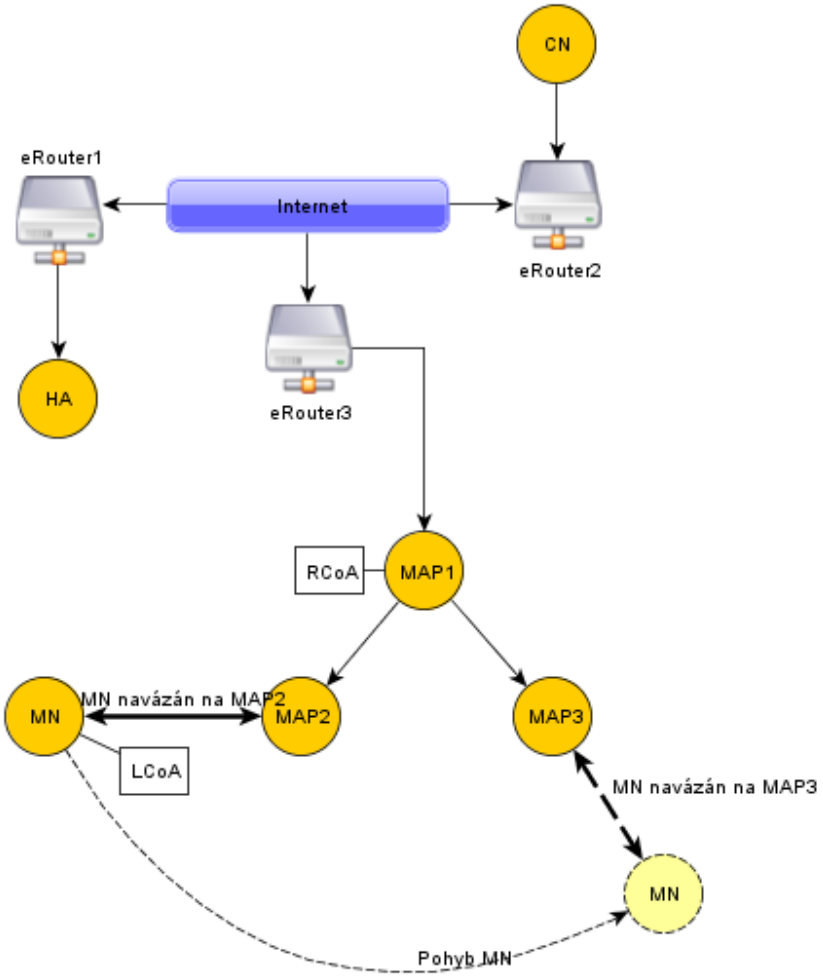
Když MN vstoupí do cizí sítě, získá adresu LCoA. LCoA vznikne z prefixu sítě a rozhranní MN. Princip získávání této adresy je standardní, jako pro každý jiný uzel v IPv6 (autokonfigurace, zprávy router advertisement a solicitation). Topologicky výše nad MN je MAP2 a právě z prefixu této domény je vytvořena LCoA. [3]

Registrace u MAP a HA

MN přijme ohlášení směrovače s volbou MAP (popsáno v 4.1.1). V ohlášení směrovače jsou obsaženy informace o dostupných MAP a o jejich parametrech dostupnosti (priorita, vzdálenost od MN). MN jako odezvu na ohlášení směrovače pošle lokální aktualizaci vazby (LBU). V této zprávě MN informuje MAP o adrese svého HA a o svojí LCoA. Pokud je v síti několik prvků MAP za sebou (jako na obrázku 4.3), posílá se tato aktualizace vazby postupně, až k topologicky nejvyššímu prvku MAP (zde od MAP1 k MAP2). Tento topologicky nejvyšší MAP také zahájí potvrzení žádosti o lokální aktualizaci vazby (LBU) (pakliže souhlasí s přístupem MN do své sítě) a zpráva potvrzení aktualizace (orig. „Binding Update Acknowledgement“) se bude šířit až k MN

(zde přes MAP2 do MN). Jednotlivé prvky MAP si uloží záznam o provedeném „bindování“ do své paměti vazeb (orig. „Binding Cache“). Jakmile je potvrzena LBU, je vytvořen tunel mezi MAP

a MN (zde mezi MAP1 a MN). Dále se posílá aktualizace vazby (BU) prvku HA. Tato zpráva obsahuje informaci o RCoA, pod kterou bude od nynějška MN dostupný[7]. RCoA vznikne z globálního prefixu sítě MAP1 a z linkového rozhranní MN. Potvrzení BU je provedeno opět zprávou BAck. V optimalizované verzi HMIPv6 pošle MN zprávu BU ohlašující dostupnost MN na adrese RCoA také na adresu CN, reakce CN bude standardní (buď spuštění RRP nebo aktualizace vazby, dle pravidel uvedených v MIPv6). [3]



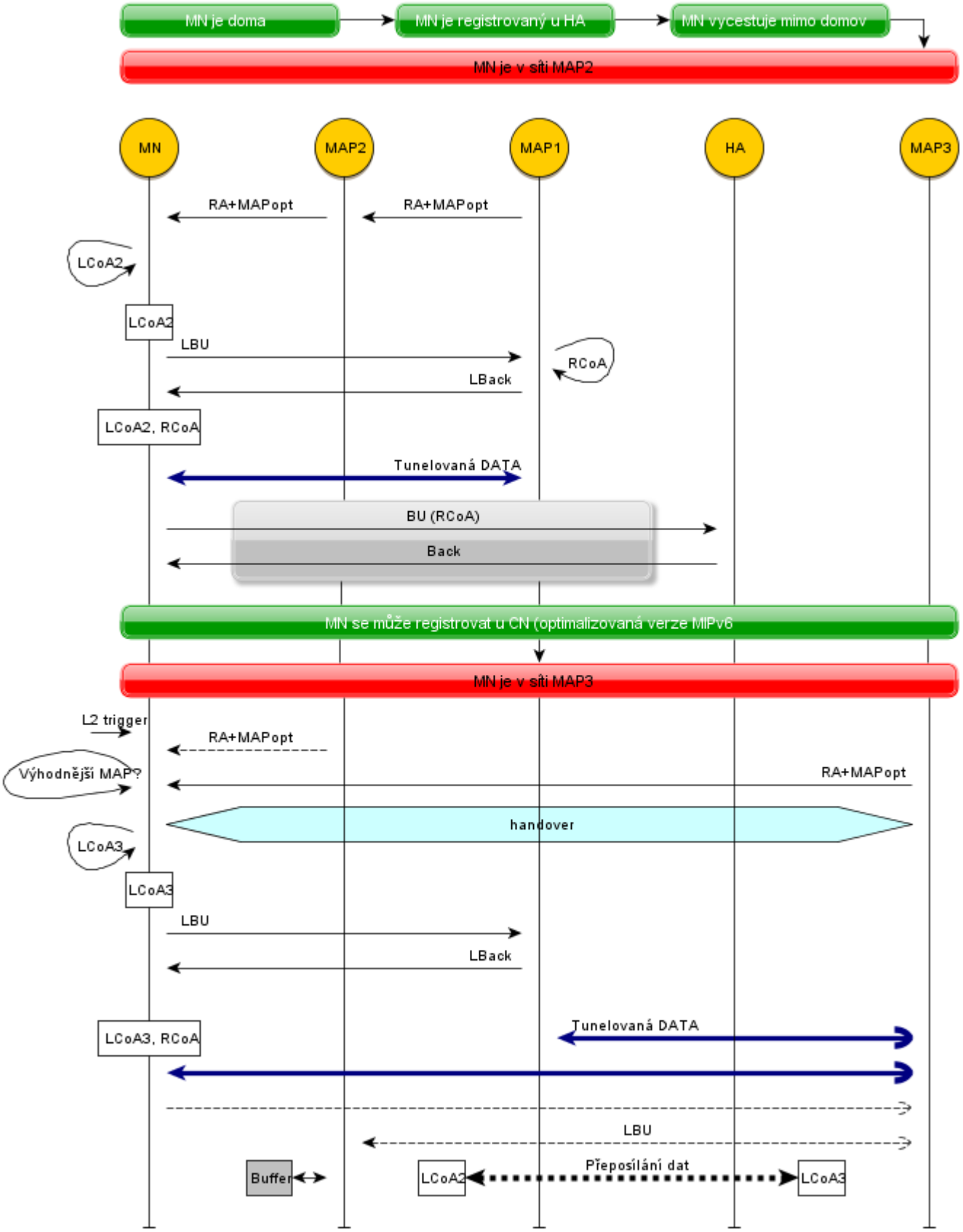
Obr 4.3: Topologie HMIPv6

Pohyb MN uvnitř sítě MAP1

Pohne-li se MN natolik, že už nemůže být spojen s aktuálním prvkem MAP (MAP2) nebo by bylo výhodnější spojit se s MAP3, pošle MN zprávu LBU na adresu nového prvku MAP (MAP3), ten ji přepošle nadřazenému prvku MAP (MAP1) a zde dojde k aktualizaci adresy LCoA, na které je aktuálně MN dostupný. Parametry spojení od MAP1 směrem k HA a CN se však nezmění a není proto nutné, měnit další parametry spojení. RCoA bude nutné měnit až v případě, že MN zcela

[7] MN bude dostupný nepřímo. To znamená, že data, která pošle CN na adresu RCoA, budou uzlem MAP přeposlána na adresu LCoA. MAP díky předchozím krokům registrace MN ví, jakou má MN LCoA.

opustí síť MAP1. Volitelně je doporučováno, aby si MN pomocí LBU vyžádal, při přechodu do nové domény MAP3, přeposlání paketů ze staré LCoA2 na novou LCoA3. Toto však může být zakázáno v pravidlech MAPu (administrátorem sítě).



Obr. 4.4: Zprávová výměna mezi prvky sítě při HMIPv6

4.4 Shrnutí k HMIPv6

Výhody HMIPv6

Pohybuje-li se MN v rozsáhlé přesto síti, která však patří do domény jednoho poskytovatele, není nutné měnit všechny parametry spojení mezi MN a HA. Mění se parametry spojení pouze uvnitř této rozsáhlé sítě, mobilita je zajištěna. HMIPv6 je pro HA a CN zcela transparentní.

Nevýhoda HMIPv6

Opět zcela neřeší problém při komunikaci vyžadující „realtime“ přenos dat, protože při změně MAP při pohybu MN, dochází k prodlevě spojené s registrací u nového MAP. Tato registrace však proběhne rychleji, než jak by probíhala v MIPv6. Až F-HMIPv6 umožňuje přenos „realtime“ dat, jako například videokonferenci.

5 RYCHLÝ HANDOVER V HIERARCHICKÉ SÍTI [F-HMIPv6]

Technika rychlého handoveru v HMIPv6 doposud nebyla standardizována specifikací RFC. Zatím existují pouze návrhy ve formě „draft“ dokumentů. Je proto možné, že se v budoucnu změní některé záležitosti, týkající se podpory rychlého handoveru uvnitř HMIPv6.

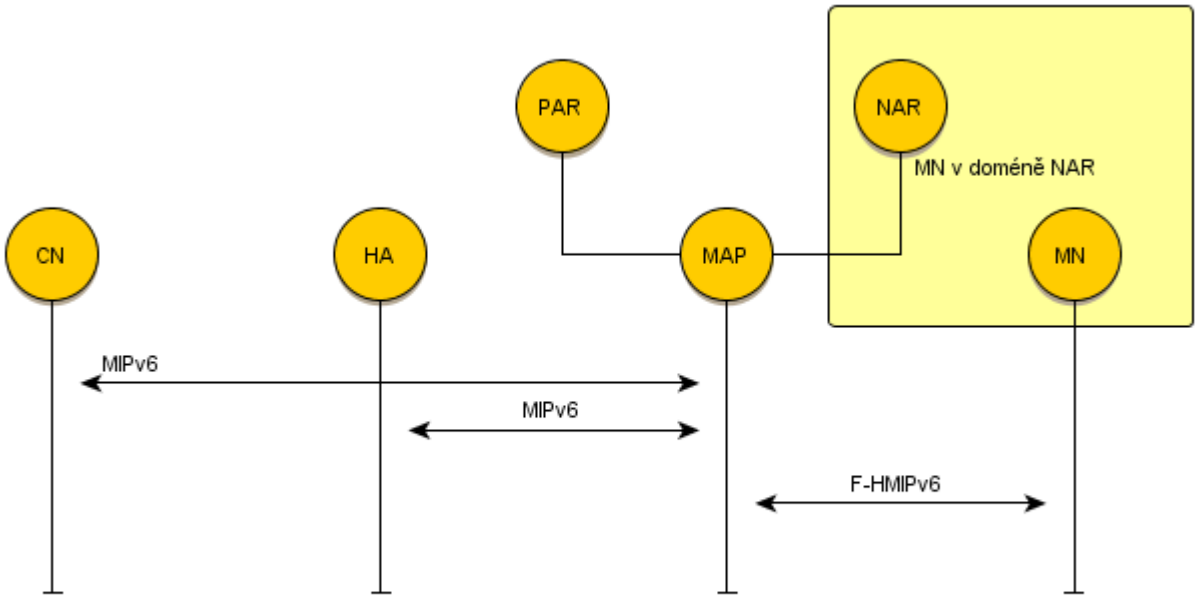
F-HMIPv6 kombinuje dvě předchozí metody, používá techniku rychlého handoveru v hierarchické mobilitě. Výsledkem je řešení, které již umožňuje předání mobilního uživatele (MN) mezi sítěmi bez ztráty dat a snižuje objem zprávové výměny, potřebné při komunikaci MN s korespondentem (CN) a domácím agentem (HA).

Topologie je přebrána z HMIPv6. Je stejná, jako na obrázku 4.3, pouze s tím rozdílem, že routery MAP2 a MAP3 budou plnit funkci PAR a NAR z FMIPVv6. MAP1 zde zůstává jako kotevní bod [5]. MN, který se dostane do domény umožňující F-HMIPv6, se nejprve připojí k PARu. MN se může bezstarostně pohybovat v síti dále, protože díky technice rychlého handoveru bude včas vědět, kam se připojit, až ztratí připojení k PARu. Připojí se k NARu. Detaily zprávové výměny jsou popsány v kapitole 5.3.

5.1 Implementace FMIPv6 do HMIPv6

Je možné zavést podporu rychlého handoveru do HMIPv6 přímo. MN se připojí k PARu a data mu budou tunelována z adresy RCoA od MAPu. Bude-li výhodnější se po čase připojit k NARu (MN se vzdaluje od PARu), vyšle MN zprávu FBU na adresu PARu a ten vytvoří tunel mezi PARem a NARem. O data se pak MN přihlásí zprávou UNA v síti NARu. Toto řešení však sebou nese nevýhody. Není efektivní data posílat na adresu PARu za účelem tunelování k NARu, protože půjdou zpět ve směru PAR-MAP-NAR. V literatuře [6] je efektivita tohoto řešení testována.

Byla proto zavedena optimalizace, která mění směr tunelování zpráv. Při inicializaci handoveru budou data tunelována z MAPu přímo na adresu NARu, kde se o ně MN přihlásí po vstupu do sítě [5]. Implementace FMIPv6 v rámci HMIPv6 je taková, jakou znázorňuje obrázek 5.1.



Obr. 5.1: Implementace FMIPv6 v rámci HMIPv6.

5.2 Zprávy a jejich hlavičky definované v rámci F-HMIPv6

Dle [5], není potřeba zavádět pro F-HMIPv6 nové zprávy. Všechny zde používané zprávy, jsou již definovány, v rámci MIPv6 (RFC 3775), HMIPv6 (RFC 5380), nebo FMIPVv6 (RFC 5268).

Oproti FMIPVv6 se mění pouze adresy příjemců a odesílatelů těchto zpráv [5]. Změny shrnuje tabulka 5.1.

Tab. 5.1: Nové adresy odesílatelů a příjemců zpráv v F-HMIPv6

Zpráva	**Použití zprávy pro F-HMIPv6**		**Použití zprávy pro FMIPVv6**	
	Zdroj	**Cíl**	**Zdroj**	**Cíl**
RtSolPR	MN	MAP	MN	PAR
PrRtAdv	MAP	MN	PAR	MN
FBU	MN	MAP	MN	PAR
FBAck	MAP	MN (přes PAR/NAR)	PAR	MN (z PARu/z NARu)
HI	MAP	NAR	PAR	NAR
HACK	NAR	MAP	NAR	PAR

Aby MN poznal, že se pohybuje uvnitř sítě, která podporuje F-HMIPv6, je modifikována zpráva ohlášení směrovače ve volbě MAP (orig.: „Router Advertisement, MAP Options“). V hlavičce zprávy pro objevení mapu (obrázek 4.1) je nový bit „F“. Bit „F“ je umístěn vlevo od bitu „R“. Pakliže je „F“ nastaven, MN pozná, že se pohybuje uvnitř F-HMIPv6 sítě. [5]

5.3 Popis síťové komunikace v doméně F-HMIPv6

Popišme si komunikaci zprávové výměny mezi prvky MN, PAR, MAP a NAR. Topologie je totožná jako na obrázku 4.3, pouze s tím rozdílem, že routery MAP2 a MAP3 budou plnit funkci PAR a NAR. Obrázek 5.2 shrnuje tuto komunikaci.

MN je doma a poté vycestuje

Začátek komunikace je totožný s MIPv6. MN se nejprve zaregistruje u HA v domácí síti pomocí aktualizace vazby (viz. 2.3.3 a obrázek 2.7). Jakmile MN vycestuje mimo domácí síť a pomocí zpráv Router Advertisement (ohlášení směrovače (viz. 2.2.2)) zjistí, že se dostal do jiné sítě (cizí), provede registraci u přístupového bodu PAR. [1]

Registrace u PARu

MN získá standardními mechanizmy, popsanými v rámci IPv6 adresu IP a další parametry, které potřebuje pro úspěšné připojení do domény PARu. Jedná se o metody autokonfigurace a objevování sousedů (viz. 2.2.1 a 2.2.3). Výsledkem je získání adresy PCoA. [1]

Registrace u MAP a HA

MN jako odezvu na Router Advertisement pošle lokální aktualizaci vazby (LBU) na adresu MAPu. V této zprávě MN informuje MAP o adrese svého HA a o svojí LCoA. MAP odešle potvrzení žádosti o lokální aktualizaci vazby (LBAck) na adresu MN (pakliže souhlasí s přístupem MN do své sítě). MAP si uloží záznam o provedeném „bindování“ do své paměti vazeb (orig. „Binding Cache“). MAP bude od této chvíle tunelovat všechny pakety přicházející z adresy HA na adresu MN (PCoA). Tunelovány budou také pakety opačného směru. Dále se posílá aktualizace vazby (BU) prvku HA (v obrázku 5.2 není znázorněno potvrzení BAck, ale ve skutečnosti proběhne). Pomocí této zprávy MN informuje HA, že je dostupný na adrese MAPu (RCoA). [3, 5]

Informace o dosažitelných handoverech.

Nyní následuje výměna zpráv v rámci FMIPVv6. Popis je totožný jako v kapitole 3.2.1-Informace o dosažitelných handoverech s tím rozdílem, že zprávy RtSolPR a PrRtAdv se vyměňují mezi prvky MAP1 a MN. [2, 5]

Handover

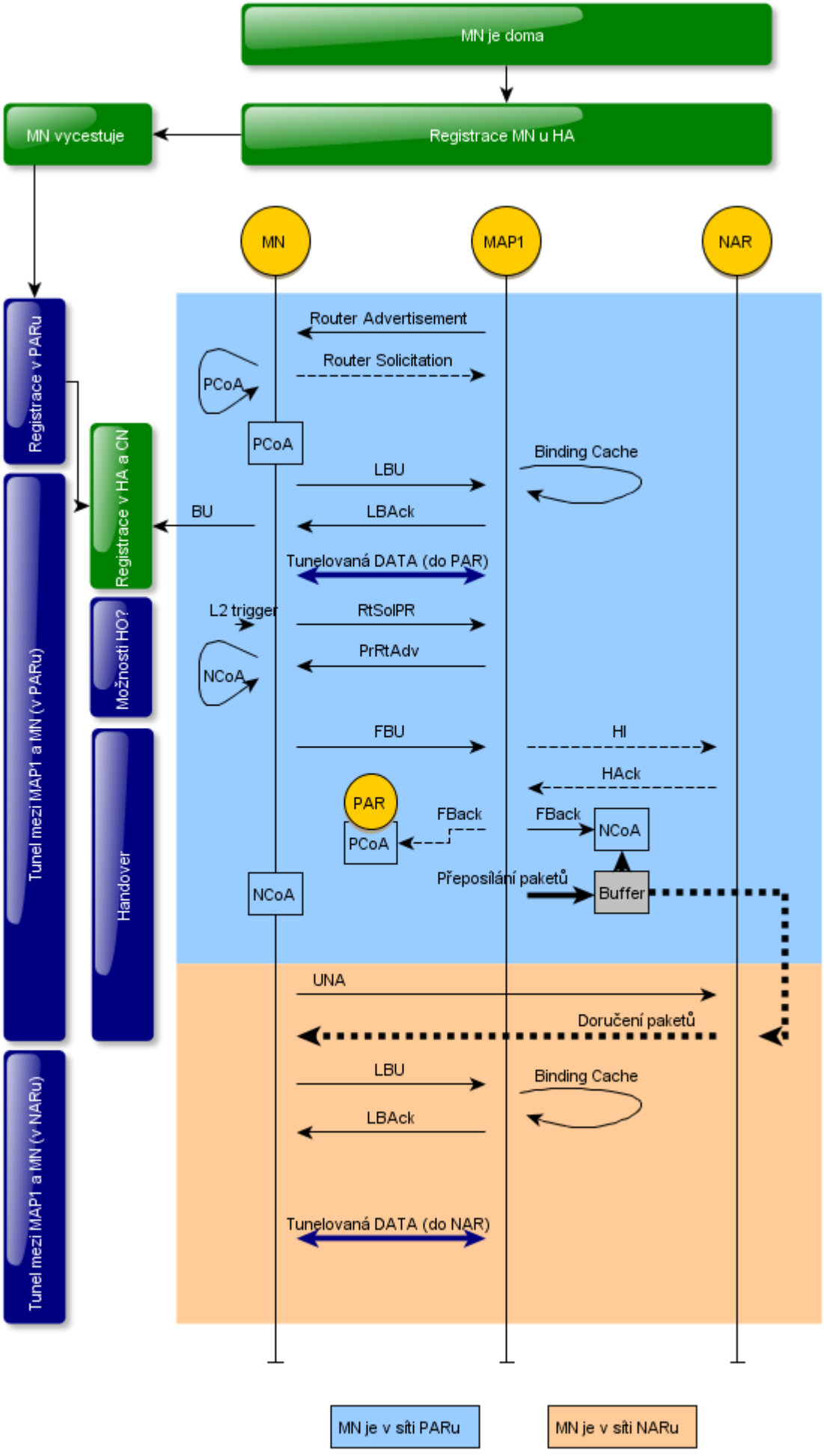
Následuje handover. MN se chce přepojit na NAR, protože je to pro něj výhodnější. Zprávová výměna je již popsána v rámci kapitoly 3.2.1. Mění se pouze adresy příjemce/odesílatele zprávy. Tyto adresy lze vyčíst z tabulky 5.1 nebo přímo z obrázku 5.2.

Tunel mezi MAP1 a MN (v NARu)

Jakmile se MN dostane do sítě NARu a jsou mu doručeny pakety, které se během handoveru ukládali v bufferu, aktualizuje MN svojí vazbu s MAP1 (opět pomocí LBU, odpovědí je LBAck. Tímto MN ohlásí MAPu svojí novou pozici (NCoA). Od této chvíle budou data mezi HA, CN – MN tunelována z RCoA do NCoA a naopak. MN již znovu nemusí aktualizovat vazbu s HA a CN, protože je stále dostupný na stejné adrese RCoA. [1, 5]

5.4 Shrnutí k F-HMIPv6

Tato technika vylepšuje HMIPv6 o možnost handoveru bez ztráty dat. Vyžaduje použití bufferů na straně přístupových bodů, ale na druhou stranu již umožňuje například přenos videokonference i v okamžiku přechodu MN mezi sítěmi. Předchozí techniky (MIPv6, HMIPv6) toto nedokáží. Další výhodou je snížení počtu zpráv, které se přenáší mezi MN - HA, CN. MN již nemusí aktualizovat vazbu tak často, protože změna přístupového bodu v doméně MAP vyžaduje pouze lokální aktualizaci vazby u MAPu.



Obr. 5.2: Zprávová výměna mezi prvky topologie F-HMIPv6

6 Úvod do vývojového prostředí OPNET

OPNET umožňuje simulaci velkého množství komunikačních protokolů různých sítí (LAN, MAN, WAN, GSM, a mnohé další). OPNET poskytuje několik editorů, které umožňují vytváření modelů na různých úrovních abstrakce. Základní jsou tyto:

6.1 Editor projektu

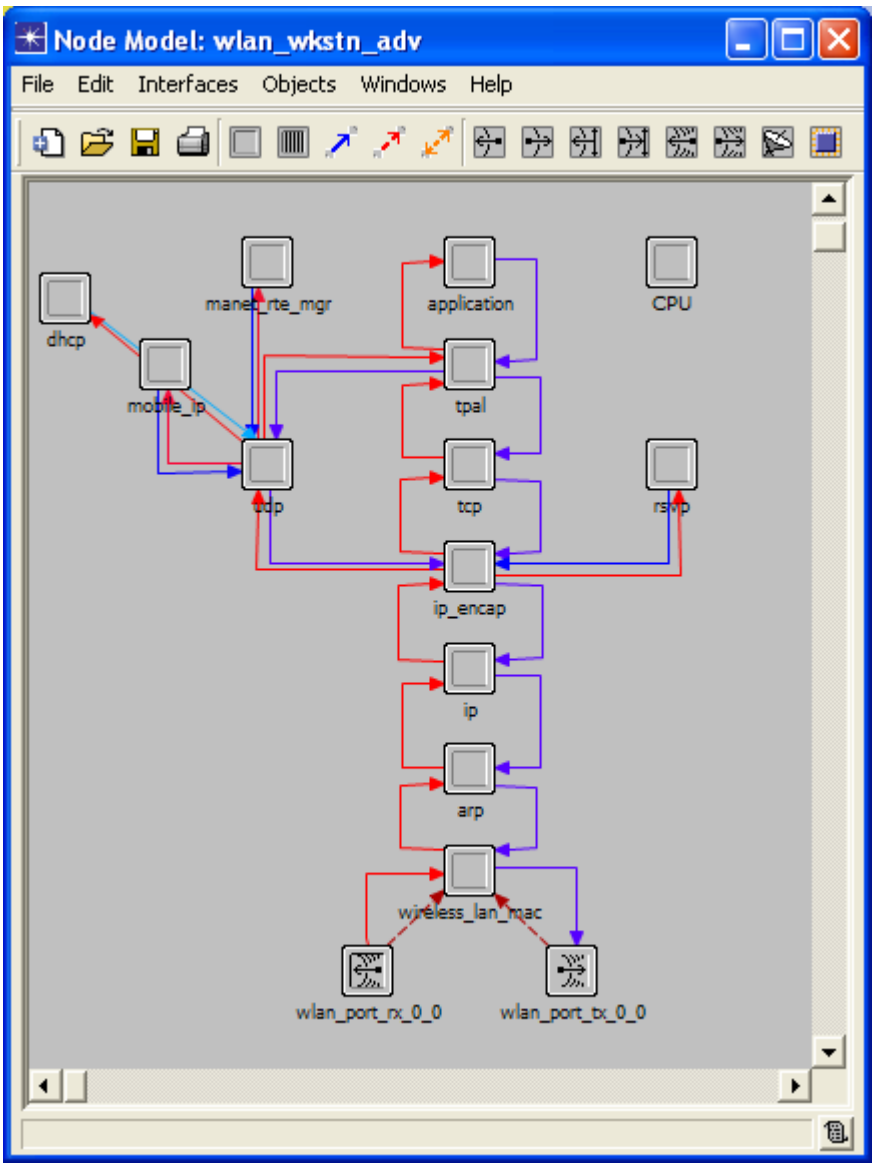
Jedná se o editor projektu, kde navrhujeme topologii sítě, na které budeme provádět simulaci. V tomto editoru můžeme použít již definované prvky sítě, jako jsou routery, switche, servery nebo i prvky UMTS (node b) a propojit je linkami. Zvolíme parametry simulace a výsledkem budou zvolené charakteristiky (grafy, směrovací tabulky, …). Příklad tvorby v project editoru projektu je uveden v kapitole 9.

6.2 Editor uzlu

Editor uzlu umožňuje propojování jednotlivých bloků kódu do větších celků. Tyto bloky kódu jsou tvořeny v editoru procesu (kapitola 6.3) a propojením těchto bloků definujeme prvek, který použijeme v editoru projektu. Propojení může být realizováno paketovým tokem nebo i jinými způsoby, které jsou popsány v produktové dokumentaci OPNETu [14].

Obrázek 6.1 ukazuje propojení jednotlivých modulů pro zajištění funkcí v rámci TCP/IP mobilního uživatele. Moduly dohromady zajišťují funkci celého uzlu. Vidíme, že od zdola jsou zajišťovány funkce pro připojení k bezdrátové síti prostřednictvím dvou rozhranní. Jedno slouží pro vysílání a druhé pro příjem dat [15]. Blok *wirless_lan_mac* a *arp* obstarávají funkci zařízení na úrovni MAC adres a zajišťují funkce spojené se získáním MAC adres okolních prvků v síti. Blok *ip* se stará o adresaci na úrovni síťové (vrstva 3) vrstvy TCP/IP modelu. *Ip_encap* zajišťuje zpracovávání paketů. Provádí zabalování a rozbalování paketů, zajišťuje spolupráci s moduly *tcp, udp, rsvp*. Nejvýše v editoru uzlu jsou uvedeny bloky sloužící aplikační vrstvě.

Nás bude v další práci zajímat především modul *mobile_ip* a modul *ip*. Právě zde je zajištěna podpora mobility v rámci síťové vrstvy. OPNET podporuje MIPv4 a MIPv6, včetně mechanizmů pro optimalizaci směrování a ověření směrování (tzv. return mutability test). [14]



Obr. 6.1: Editor uzlu – ukázka propojení jednotlivých bloků

6.2.1 Komunikace mezi uzly a procesy

Mezi uzly nebo přímo mezi procesy je možné komunikovat prostřednictvím přerušení. OPNET nabízí více možností. Nejvyužívanější je komunikace prostřednictvím šíření paketů, volání procesu (op_intrpt_schedule_process(), op_pro_invoke()) a vzdáleným přerušením (op_intrpt_schedule_remote()).

Všechny tyto funkce potřebují vstupní parametry, které jsou definované v nápovědě opnetu. Někdy je nutné využít dalších mechanismů pro získání těchto parametrů. Například potřebuji-li získat *Objid* uzlu jiného, než ze kterého chci komunikovat, lze využít objevení registrace procesu (oms_pr_process_discover()) nebo získání *Objid* ze jména (op_id_from_name()). První způsob je výhodnější, jelikož mohu prostřednictvím registrace sdílet i další atributy procesu (zpřístupnění atributů pomocí oms_pr_attr_get()).

Většina druhů přerušení také umožňuje šíření doplňujících údajů prostřednictvím ICI. Postup, jak lze definovat ICI je následující [14]:

1. Vytvořím strukturu ici v editoru ici (New/ICI format) a uložím ji do adresáře s vlastním projektem.
2. Vytvořím ici: op_ici_create().
3. Naplním ici atributy, které chci přerušením přenášet: op_ici_attr_set().
4. Nainstaluji ici. Ici se automaticky přidá k odchozímu přerušení: op_ici_install().
5. Zruším ici, pakliže již nepotřebuji přenášet informace: op_ici_destroy().

Při vyzvedávání info z ici musím získat ukazatel na ici (op_intrpt_ici()) a načíst údaje z ici (op_ici_attr_get()). Využití je ukázáno při implementaci FMIPv6 v kapitole 8.1.1.

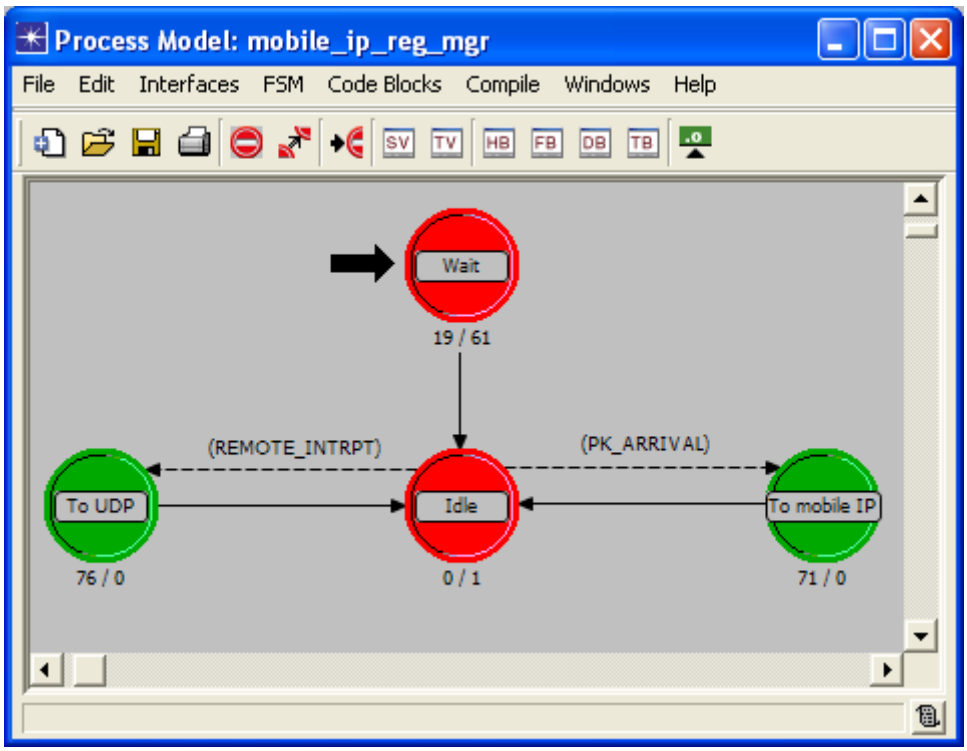
6.3 Procesní model

Nebo také editor procesů, operuje na nejnižší úrovni z uvedených editorů. Poskytuje nám informace o tom, jak konkrétně funguje zvolený modul. Poklikáním na modul *mobile_ip* z obrázku 6.1 se dostaneme na úroveň stavového diagramu. Tento je charakterizován jednotlivými stavy a jejich přechody, například tak, jak je znázorněno na obrázku 6.2.

Do jednotlivých stavů se vkládá kód v jazyce C/C++. OPNET obsahuje spoustu vlastních datových typů a také funkcí, které nejsou modifikovatelné, protože se jedná přímo o funkce jádra simulačního programu (kernelu).

Stav se skládá ze vstupní (enter executive) a výstupní (exit executive) části. Kód vykonaný na vstupu stavu je vykonán při příchodu proceesu do tohoto stavu. Například v obrázku 6.2 vidíme, že při příchodu do stavu *to_mobile_IP* ze stavu *Idle*, bude provedeno 71 řádků kódu. Výstupní část kódu je provedena po příchodu události (přerušení) u nevynuceného stavu nebo obecně před opuštěním tohoto stavu prostřednictvím přechodu do jiného stavu. Po vykonání kódu ve výstupní části stavu následuje vyhodnocení podmínek, které jsou definované pro přechod do následujícího stavu a provedení přechodu do stavu s pravdivou podmínkou pro tento přechod. Typ přechodu (vazby) mezi stavy se označuje jako podmíněná nebo podmíněná. Právě v závislosti na použití resp. Nepoužití pomínky.

Stav může být reprezentován zeleným (unforced) nebo červeným (forced) objektem. Zelený je takzvaně vynucený. Po vykonání kódu uvnitř tohoto stavu následuje okamžitý přechod do stavu následujícího (ke kterému vede šipka). Červený stav je takzvaně nevynucený. Po provedení kódu ve vstupní části stavu se čeká na příchod události (přerušení), aby mohla být vykonána také výstupní část kódu stavu a mohlo dojít k přechodu do následujícího stavu.



Obr. 6.2: Procesní model – ukázka propojován stavů

6.3.1 Stavové proměnné

Nejdříve se definují atributy v nadřazeném (root) procesním modelu v *Interfaces/Model Atribute* a zadefinují se jím odpovídající stavové proměnné. Mohou zde být definovány také proměnné u kterých požadujeme platnost v celém modulu. Čtení z atributů a ukládání do stavových proměnných je realizováno uvnitř procesního modelu pomocí funkcí.

6.3.2 Blok hlaviček

V tomto bloku jsou definovány různé konstanty a připojovány hlavičkové soubory, které používáme v modelu.

6.3.3 Blok funkcí

Obsahuje jednotlivé funkce, které mohou být volány mezi sebou v rámci tohoto bloku nebo mohou být volány z procesního modelu, z jednotlivých stavů.

II) PRAKTICKÁ ČÁST

7 SIMULACE MIPV6 V PROGRAMU OPNET

V této kapitole si ukážeme simulaci sítě podporující základní verzi mobility, tedy MIPv6. Simulace bude provedena v prostředí Opnet verze 14.5.

Opnet 14.5 podporuje simulace MIPv4 i MIPv6. Pro MIPv6 dokonce umožňuje optimalizovaný provoz včetně provedené „return routability" procedury. Zatím však nepodporuje pokročilé metody handoverů (FMIPv6, HMIPv6, F-HMIPv6).

7.1 Zadání

Simulace bude provedena na modelu skládajícím se z šesti sítí, kterými bude procházet mobilní uživatel (můžeme si ho představit jako uživatele s notebookem a rozhranním wi-fi). MN bude začínat svůj pohyb v domovské síti, kde bude bezdrátově připojený k přístupovému bodu. Tento přístupový bod bude plnit funkci domovského agenta, bude pojmenovaný HA. V dalších sítích bude také přístupový bod. Vždy v jedné síti jeden AP. Tyto AP budou pojmenovány FN1 až FN5. Každý AP bude mít jinou IPv6 adresu, tím bude zajištěno, že se bude v modelu vyskytovat šest různých sítí. AP budou připojeny na ethernetovou síť. Vnitřní struktura sítě se skládá ze dvou routerů a „cloudu" (mraku), který představuje internetovou síť. HA, FN1, FN2, FN3 budou připojeny na jednom routeru, který bude mrakem oddělený od druhého routeru. Na tomto druhém routeru budou připojeny zbývající FN (FN4 a FN5). Toto řešení by mělo poskytnout viditelné výsledky v simulovaných scénářích standardní MIPv6/optimalizované MIPv6. Celá síť musí být IPv6 kompatibilní, aby bylo možné zajistit podporu MIPv6.

7.2 Řešení

7.2.1 Vytváření topologie

Scénář je vytvořen v prostředí „Campus" o rozměrech 10x10 km.

Do scénáře byly vloženy následující prvky a poskládány do simulace tak, jak znázorňuje obrázek 7.1.

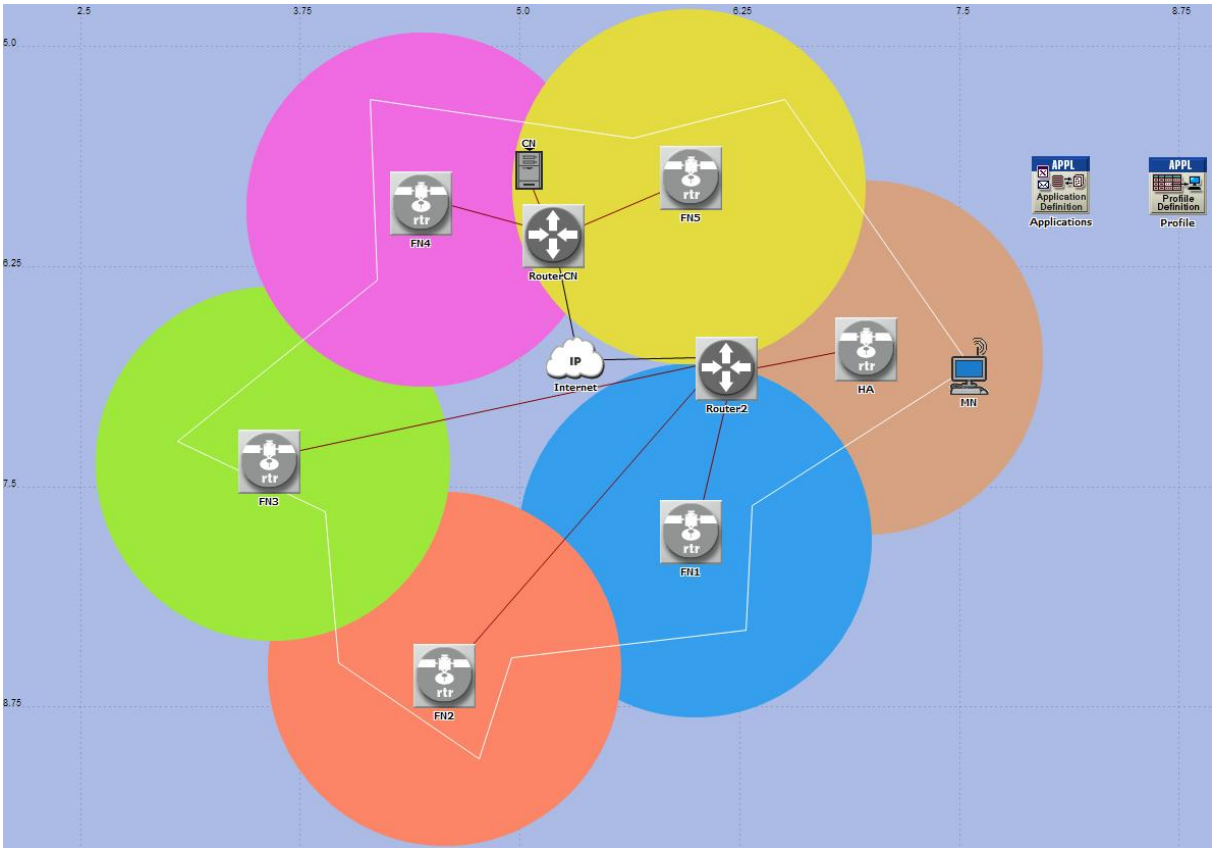
Tab. 7.1: Prvky použité v simulaci MIPv6 v programu Opnet

Název	Počet	Knihovna	Prvek
Mobilní uživatel (MN)	1	MIPv6_adv	wlan_wkstn_adv (mobile)
Korespondent (CN)	1	MIPv6_adv	mipv6_ethernet_server_adv (fixed)
Domácí agent (HA)	1	MIPv6_adv	wlan_ethernet_router_adv (fixed)

AP cizích sítí (FN1-5)	5	MIPv6_adv	wlan_ethernet_router_adv (fixed)
Router (RouterCN)	1	mobile_ip	mip_atm2_ethernet32_frelay4_slip32
Router (Router2)	1	mobile_ip	mip_atm2_ethernet32_frelay4_slip32
Mrak (Internet)	1	mobile_ip	ip8_cloud
Applications	1	internet_toolbox	Application Config
Profile	1	internet_toolbox	Profile Config

Poznámka k výběru prvků

Opnet umožňuje do simulace MIPv6 vybrat dva typy pevných serverů (pro ethernet a sériovou linku) a dokonce i mobilní nebo fixní bezdrátový server. Jako mobilní uživatel může sloužit wlan_wkstn_adv (mobile). Pro výběr routerů uvnitř sítě jsme však omezeni pouze na *mip_atm2_ethernet32_frelay4_slip32*, protože ten jediný podporuje mobilitu. Ostatní modely routerů (například ten z knihovny internet_toolbox) mobilitu nepodporují. Je to způsobeno tím, že RFC 3775, kde je MIPv6 předepsána, modifikuje hlavičky „router advertisement" a standardní IPv6 router (alespoň je to takto nastavené v Opnetu) nedokáže přenést data od CN k MN.



Obr. 7.1: Topologie MIPv6 simulovaná v prostředí Opnet.

Konfigurace modelu v prostředí Opnet se skládá z několika kroků:

- Nastavení aplikací, na obrázku je zastoupeno blokem Applications. Umožňuje nám definovat aplikace, které budou v síti spouštěny.
- Nastavení profilů, na obrázku je zastoupeno blokem Profile. V Profile můžeme nastavovat, kolikrát se spustí aplikace, jestli se bude v profilu opakovat více těchto aplikací a jaká bude posloupnost jejich spouštění. Můžeme také nastavit trvání jednoho profilu, jeho opakování a další parametry. [14]
- Nastavení jednotlivých prvků modelu: routerů, serverů, klientů, linek. Těmto prvkům je nutné v našem případě nastavit IPv6 adresy na všech použitých rozhranních a parametry mobility (popsáno níže).
- Nastavení sledovaných parametrů simulace.
- Nastavení simulace.

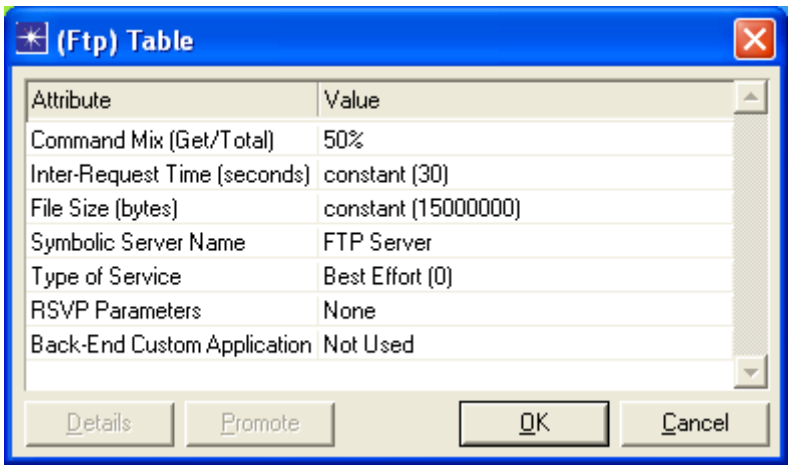
Pokud bude v textu níže použito slovní spojení „nastavením parametrů“ či podobné, myslí se tím nastavení parametrů zvoleného prvku. Do nastavování parametrů se v Opnetu dostaneme kliknutím pravého tlačítka na prvek (router, přístupový bod, link, trajektorie, …) a zvolením „Edit Attributes“. Dostupné parametry zvoleného prvku jsou uspořádány ve stromovém seznamu. Veškeré popisované akce se vztahují k verzi Opnet 14.5.

7.2.2 Konfigurace prvků topologie

Nastavení profilů a aplikací

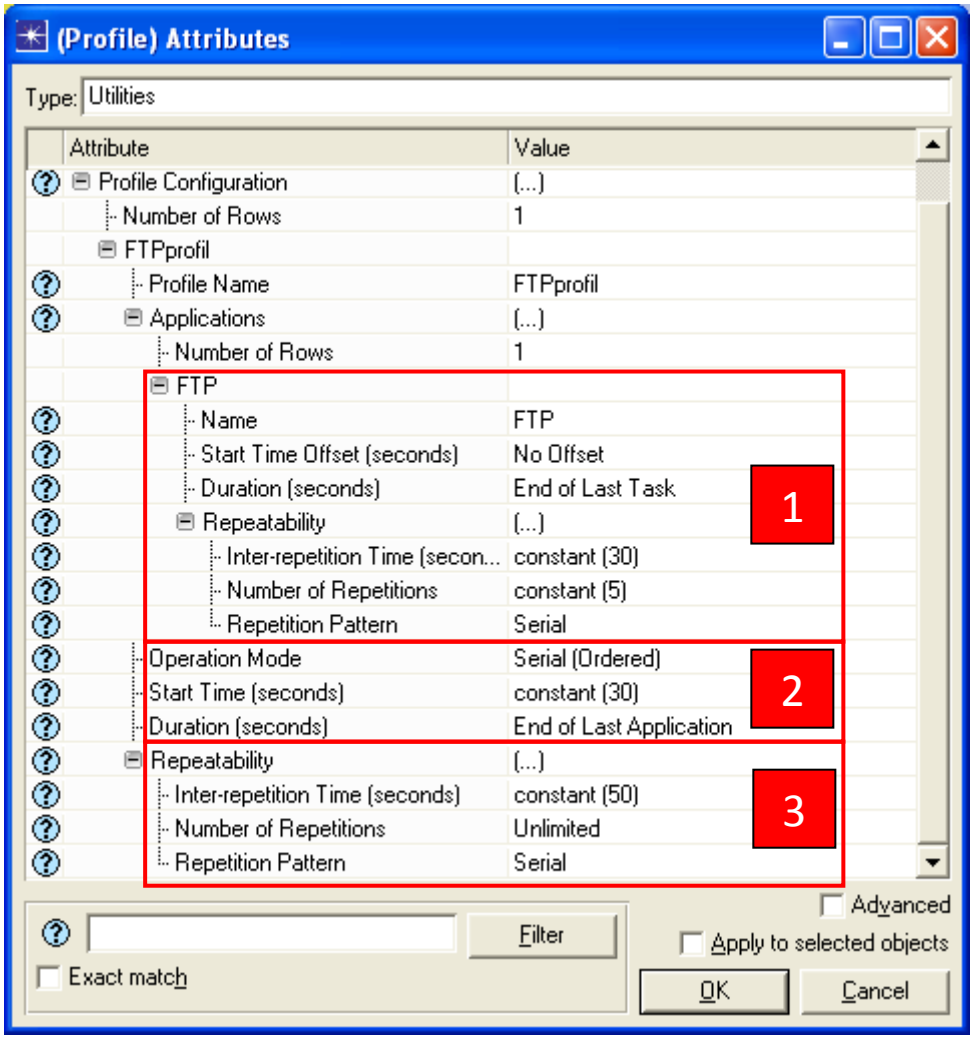
Nastavení aplikace FTP je provedeno v prvku Applications a nastavení profilu v prvku Profile (viz. Tab. 7.1).

Zvolíme *Application Definitions\Number of Rows = 1\Name = FTP\Description\Ftp* – editujeme. Nastavení zobrazuje obrázek 7.2. *Command Mix (Get/Total)* udává, jaký bude poměr mezi vysíláním příklazu get k celkovému get+put. Hodnotu nastavíme na 50%. Zpoždění mezi get a put příkazy nastavíme na 30 sekund. Původní hodnota byla exponential(360). Menší hodnotou dosáhneme většího zatížení sítě. Mezi klientem-serverem se budou posílat data o velikosti 15 MB. Nastavením menší hodnoty bychom viděli i zprávovou výměnu put/get, nás ale zajímá reakce sítě na větší provoz (zatížení).



Obr. 7.2: Nastavení aplikace FTP

Nastavení profilu provedeme v parametrech *Profile Configuration\Number of Rows = 1\Profile Name = FTPprofil* - editujeme. Nastavení je zobrazeno na obrázku 7.3



Obr. 7.3: Nastavení profilu FTP

Blok 1) popisuje, jak se bude chovat aplikace v rámci profilu.

- *Start Time Offset = No Offset*: Aplikace se bude spouštět okamžitě po spuštění profilu.
- *Duration = End of Last Task*: Délka trvání aplikace závisí na samotné aplikaci.
- V *Repeability* je definováno zpoždění spuštění následující a dalších aplikací (30 sekund) a počet jejich opakování (5x).

Blok 2) popisuje další parametry aplikace

- *Operation Mode = Serial (Ordered)*: Aplikace se budou spouštět postupně, v pořadí v jakém jsou definovány. Ještě jsou další možnosti: *Serial (Random)* = za sebou, ale náhodně a *Simultaneous* = všechny najednou (poběží paralelně).

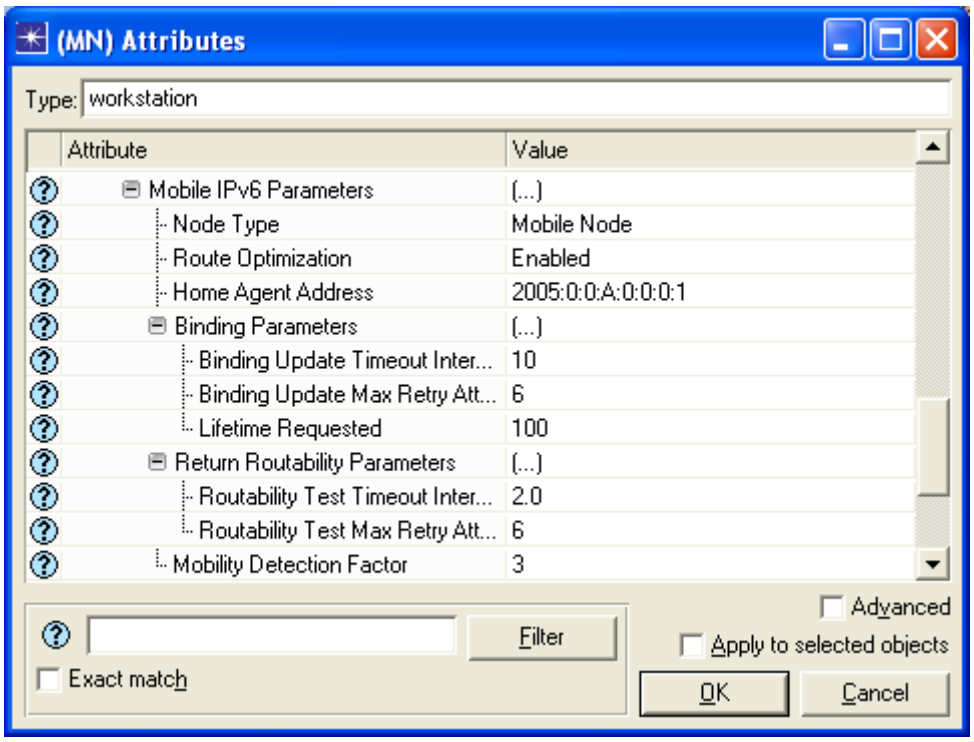
Blok 3) popisuje opakování profilů

- Mezi jednotlivými profily bude 50 sekund prodleva.
- Opakovány budou neomezeně do doby, než zkončí simulace.

Konfigurace mobilního uživatele (MN)

MN bude přistupovat k FTP serveru, který běží na uzlu CN. Je nutné mu nastavit klientský přístup k FTP serveru. Toto umožníme nastavením parametru *Applications\Application: Supported Profiles* a výběrem nakonfigurovaného profilu.

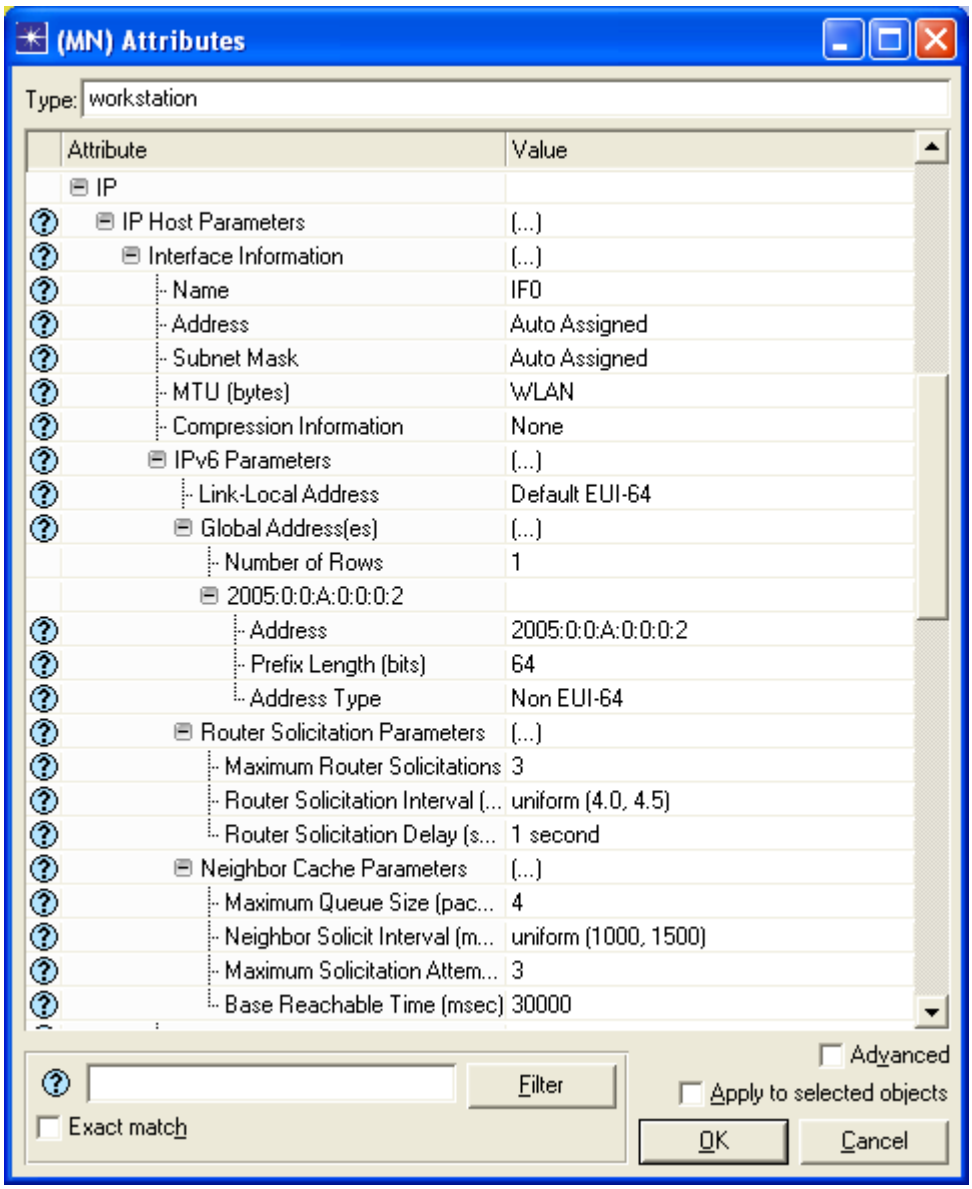
Poté je nutné nastavit parametry v *IP\Mobile IP Host Parameters\Mobile IPv6 Parameters* dle obrázku 7.4. Je důležité nastavit *Node Type* na *Mobile Node*. *Route Optimization* je nastaveno na *Disabled* v případě standardního MIPv6. V případě optimalizovaného MIPv6 je nastaveno na *Enabled*, jako v tomto případě. *Home Agent Adress* udává adresu, na které je dostupný domácí agent mobilního uživatele. Tuto není bezpodmínečně nutné konfigurovat (volba *Not Configured*) v případě, že MN začíná pohyb v domovské síti, kde si adresu HA dokáže sám získat prostřednictvím metody Objevování domácích agentů tak, jak je popsáno v kapitole 2.2.4. Parametry z kategorie *Binding Parameters* udávají, jak často se bude aktualizovat vazba mezi MN-HA resp. MN-CN v případě optimalizované MIPv6. O aktualizaci zprávy se dočtete v kapitole 2.3.3. Return Routability Parameters slouží k nastavení procedury ověření zpětného směrování. O této proceduře se dočtete v kapitole 2.2.5.



Obr. 7.4: Nastavení parametrů mobility MN

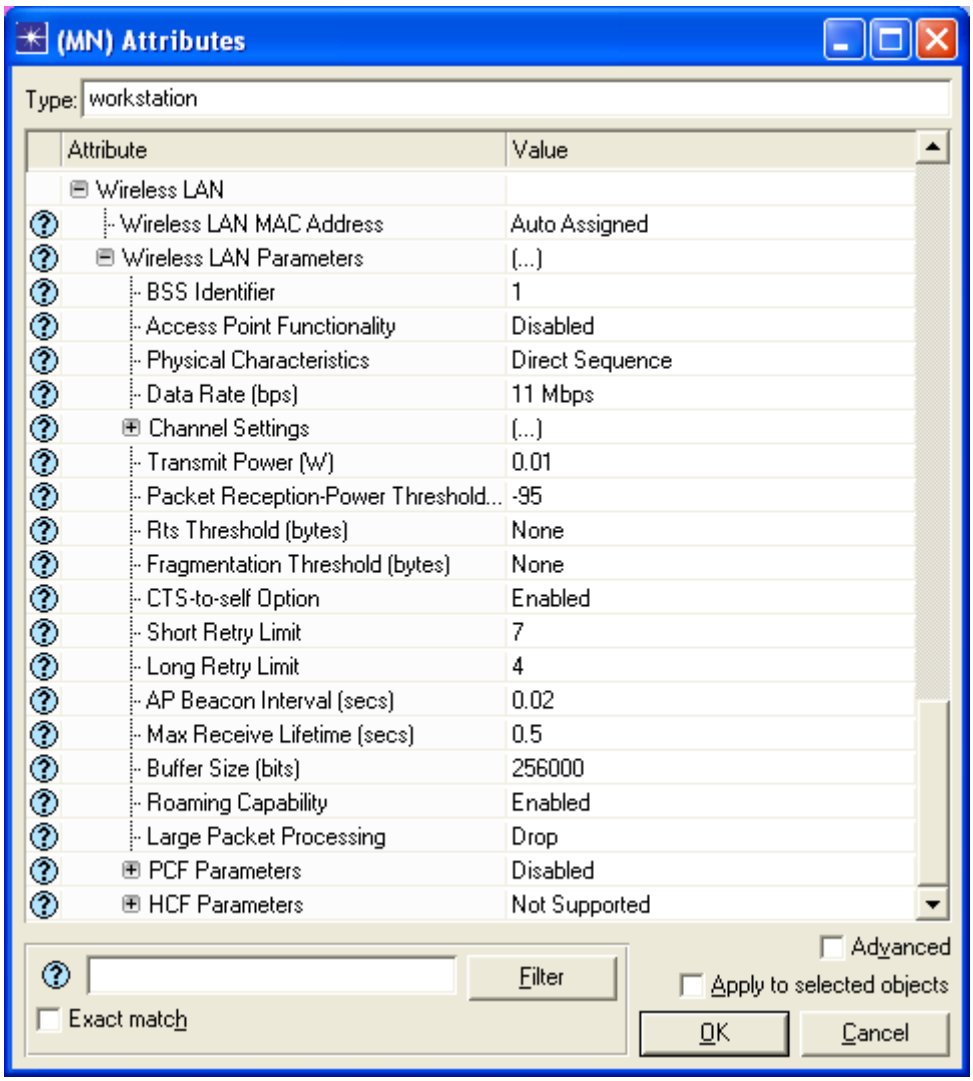
Parametry související s nastavením IP adresy a parametrů IPv6 jsou zobrazeny na obrázku 7.5. Konfiguraci IPv6 adres sítě je efektivnější nechat na volbě automatického přiřazení a ručně konfigurovat až specifické vlastnosti každého prvku. Možnost automatického přiřazení IPv6 adres je přístupná v záložce *Protocols\IPv6\Auto-Assign IPv6 addresses*. Aby tato volba byla správně funkční v bezdrátové síti, je nutné mít nastaveny *BSSID* přístupových bodů (FN1 – FN5, HA) a MN.

Na obrázku 7.5 vidíme, že MN má na rozhranní *IF0* konfigurovanou bezdrátovou síť (položka MTU je nastavena na WLAN) a výchozí adresa IP je *2005:0:0:A:0:0:0:2*. Tato adresa bude platná v domovské síti. V cizích sítích získá MN adresu dočasnou tzv CoA. Mechanismus získání CoA je popsán v kapitolách 2.5 a 2.6. Parametry z kategorie *Router Solicitation Parameters* souvisí s ohlášením MN routeru sítě, ve které se právě nachází a parametry z kategorie *Neighbor Cache Parameters* udávají, kolik sousedů si bude MN udržovat ve své paměti. MN musí znát IP a MAC adresu svých sousedů, jinak by jim nemohl posílat pakety. Zde se nastavuje, jak často se bude vysílat žádost o ohlášení směrovače (*Neighbor Solicit Interval*) a několik dalších parametrů.



Obr. 7.5: Nastavení IPv6 parametrů MN

Parametry bezdrátového spojení se nastavují ve *Wireless LAN.* Pro správnou funkci musíme nastavit *BSS Identifier*. Je nastaven na jedna. Stejnou hodnotu musí mít i HA. Důležité je také zapnout podporu roamingu (*Roaming Capability* = *Enabled*). Hodnotu vysílaného výkonu (*Transmit Power*) nastavíme o úroveň výše, než u přístupových bodů FN1 – FN5 a než u HA, abychom zajistili doručení vysílaných paketů. Zde je hodnota 10 mW.



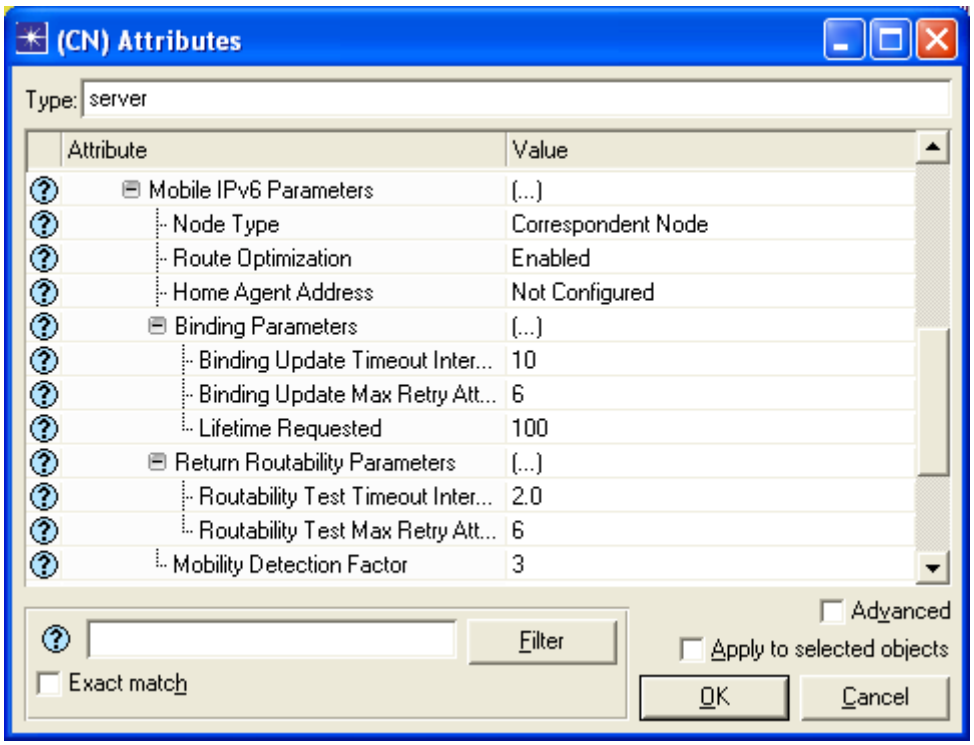
Obr. 7.6: Nastavení parametrů bezdrátové sítě pro MN

Konfigurace korespondenta (CN)

CN v této simulaci plní roli FTP serveru, ke kterému se bude MN připojovat. CN tedy nastavíme pro podporu služby FTP v parametrech *Applications\Application: Supported Services* a výběrem FTP.

Konfigurace podpory mobility pro CN se provede nastavením parametrů v *IP\Mobile IP Host Parameters\Mobile IPv6 Parameters* – dle obrázku 7.7. Je nutné mít nastaveno *Node Type* na *Correspondent Node* a *Route Optimization* na *Enabled*. Další parametry není nutné nastavovat.

Parametry týkající se IP adresace lze nastavit podobně jako pro MN. IPv6 adresa bude jiná, musí být z podsítě Routeru CN.



Obr. 7.7: Nastavení parametrů mobility CN

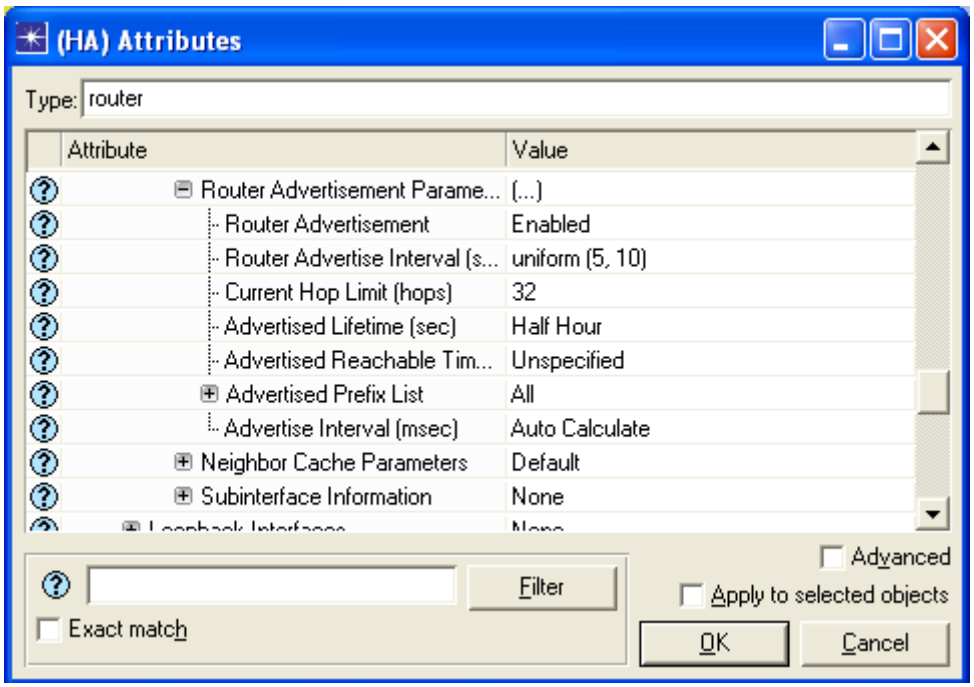
Konfigurace HA

V prvé řadě je nutné nastavit parametry v *IP\IPv6 Parameters\Interface Information*. Na rozhranní bezdrátové sítě musíme nastavit častější ohlašování směrovače, aby MN byl schopný včas detekovat, v jaké síti se nachází. Bezdrátové rozhranní máme nastaveno na IF1. *Router Advertisement Parameters* je nastaveno dle obrázku 7.8.

Dále v *IP\Mobile IP Router Parameters\Mobile IPv6 Parameters* musí být na bezdrátovém rozhranní (IF1) nastaven *Interface Type* na *Home Agent*.

Jako poslední je třeba nastavit parametry bezdrátového přenosu. Tyto se nacházejí ve *Wireless LAN\Wireless LAN Parameters*. Nastavení se od nastavení MN liší v položkách *Access Point Functionality*, *Transmit Power* a *Roaming Capability*. Nastavení je následující:

- *Access Point Functionality* = *Enabled*: HA plní na IF1 rozhranní funkci přístupového bodu,
- *Transmit Power* = *0,005* W: Výkon byl zvolen s ohledem na překrývání oblastí jednotlivých sítí, tak aby bylo možné přepojení na cizí síť.
- *Roaming Capability* = *Disabled.*



Obr. 7.8: Nastavení parametrů ohlášení směrovače

Konfigurace FN1 – FN5

Stejně jako u HA, je nutné nastavit častější ohlašování směrovače.

Není nutné konfigurovat podporu mobility, jako tomu bylo u CN, MN a HA.

Je nezbytné nastavit BSSID jednotlivých bezdrátových sítí. Nastavení je provedeno ve *Wireless LAN\Wireless LAN Parameters\BSS Identifier*. FN1 má přidělen identifikátor *2*, FN2 má *3*, atd. (FN5 má BSSID = *6*). Ostatní parametry WLAN jsou stejné jako u HA.

Konfigurace vnitřních routerů

Routery uvnitř sítě není nutné ručně konfigurovat. Routery musí být z knihovny mobile_ip. Routery jsou spojeny s „cloudem“ (mrakem) sériovou linkou.

Konfigurace mraku

Mrak v naší síti představuje Internet. Pakety, které mrakem procházejí, jsou zpožděny o 0,5 sekund. Nastavení je provedeno v položce *Packet Latency* v parametrech mraku.

Automatické přidělení IPv6 adres

Po nastavení všech parametrů, je pro vytvoření funkční topologie sítě nutné, aby se jednotlivým rozhranním prvků v síti přidělila IPv6 adresa. Možnost automatického přiřazení IPv6 adres je přístupná v záložce *Protocols\IPv6\Auto-Assign IPv6 addresses*.

Nastavení trajektorie

MN se bude v síti pohybovat dle zvolené trajektorie. V obrázku 7.1 je trajektorie znázorněna bílou barvou. Je nastaveno, aby se MN pohyboval stálou rychlostí 15 km/h. MN začne svůj pohyb

v síti s HA a bude procházet přes sítě FN1 až FN5 a vrátí se zpět k HA. Celou trasu MN urazí za 55 minut.

7.2.3 Nastavení simulace

Parametry simulace obou scénářů jsou následující:

- délka = 1 hodina,
- počet hodnot/statistika = 1000,
- spouštění simulace = optimalizované.

7.3 Výsledky simulace

Data byla exportována z programu Opnet do Excelu. Na následujících stránkách jsou graficky znázorněny nejdůležitější poznatky.

Výsledek 1 (obrázek 7.9)

První graf znázorňuje, jak jsou zpožděny data, která putují od mobilního uživatele ke korespondentovi. Modrá křivka je pro standardní MIPv6, červená pro optimalizovanou verzi. Druhý graf znázorňuje handover mobilního uživatele k jednotlivým přístupovým bodům v sítích, kterými prochází.

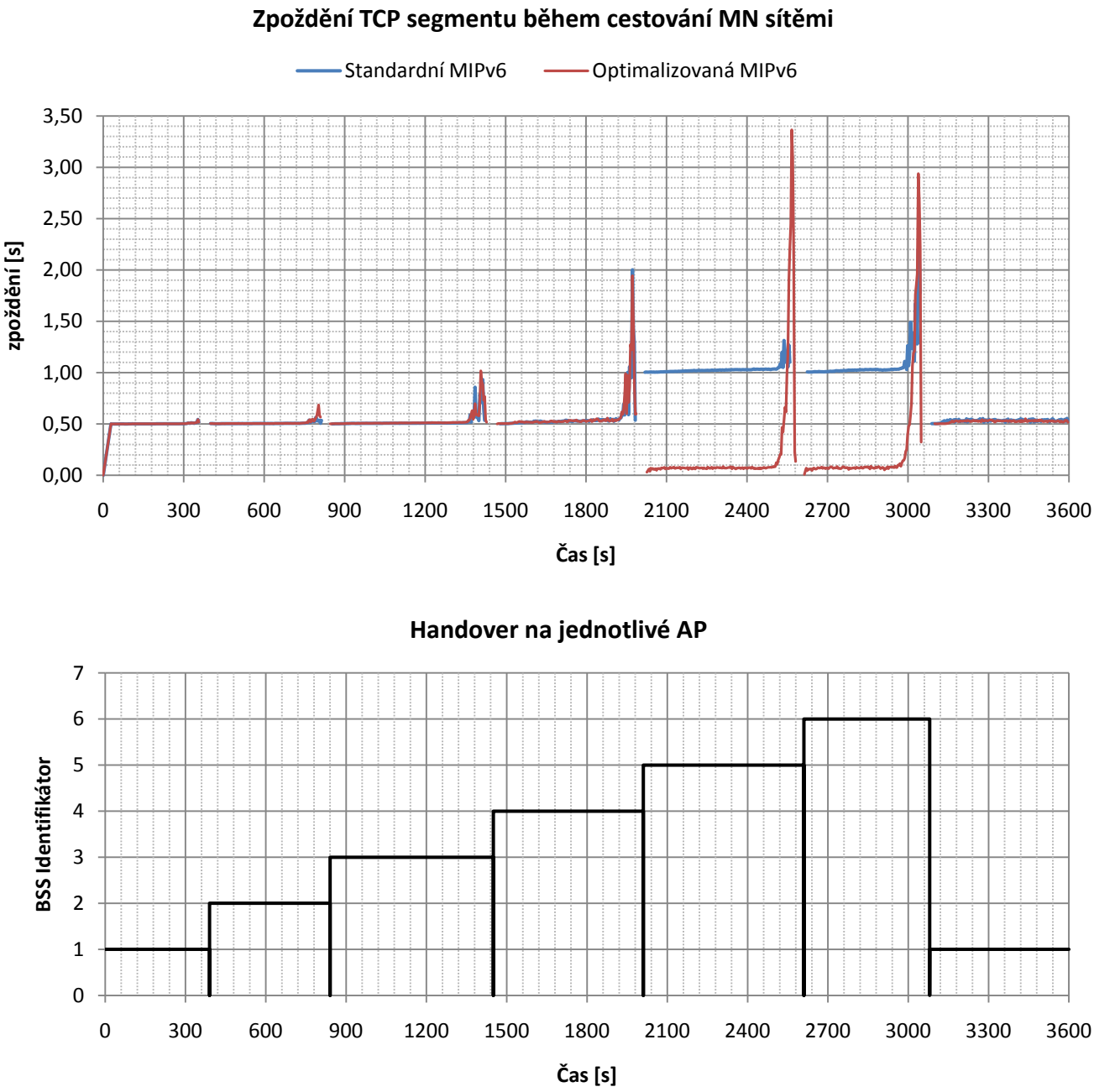
Zpoždění vzniká vlivem přenosu dat na velkou vzdálenost. Toto zpoždění je uměle nastaveno na mraku, činní 0,5 sekundy. Data, která putují k resp. od přístupových bodů FN4 a FN5 přes mrak, jsou o tento čas zpožďována. Korespondent je oddělen od HA mrakem, proto ve standardní MIPv6 dochází ke zpoždění 0,5 sekund v sítích FN1, FN2, FN3. V sítích FN4 a FN5 je u standardní MIPv6 zpoždění dokonce 1 sekunda, protože data musí procházet od CN k HA (1x přes mrak) a od HA k MN zpět do sítě FN4 resp. FN5 (2x přes mrak). U optimalizované verze je zpoždění v sítích FN1, FN2 a FN3 stejné, ale v FN4 a FN5 není žádné (reálně nemůže nastat nulové zpoždění, ale bylo by minimální). Optimalizovaná verze MIPv6 umožňuje, aby si CN udržoval informaci o aktuální poloze MN. Děje se tak prostřednictvím aktualizace vazby, kterou MN posílá CN a přenos dat mezi MN-CN pak může probíhat optimálně. Na rozdíl od standardní verze MIPv6, kdy bude přenos dat probíhat mezi CN-HA-MN.

Výsledek 2 (obrázek 7.10)

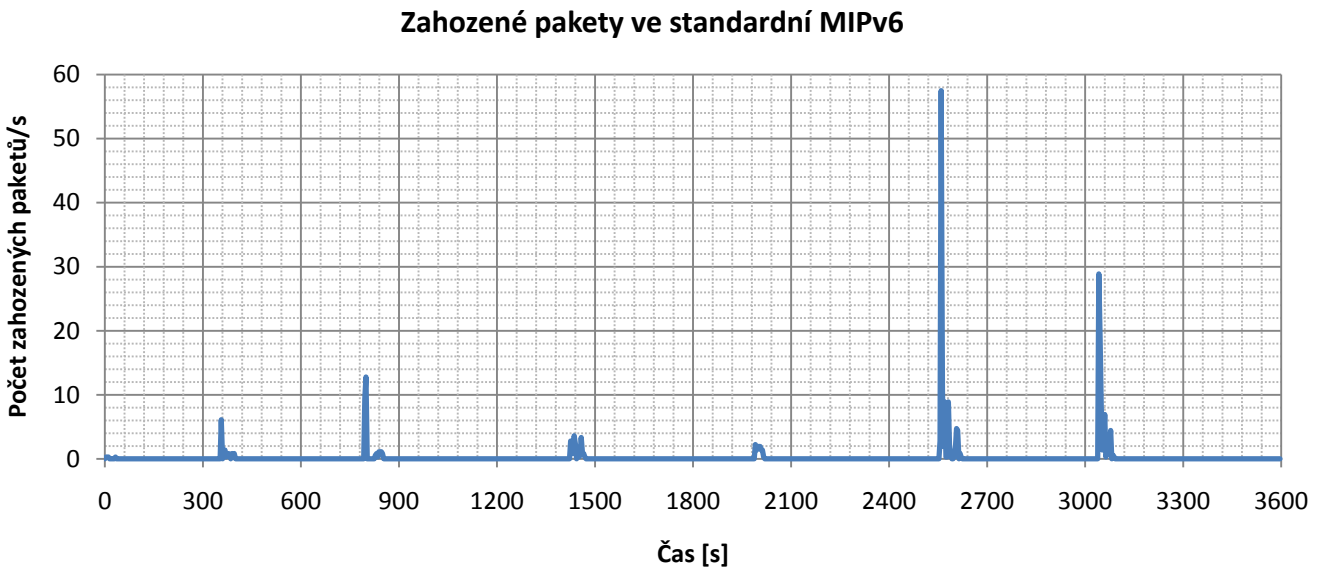
Vlivem přecházení MN mezi sítěmi dochází u MIPv6 vždy ke krátkému přerušení toku dat. Je to způsobeno tím, že mobilní uživatel opouštějící síť, ztrácí svojí aktuální IP adresu a než získá novou, uplyne krátký čas. Tento čas závisí na tom, jak často vysílají přístupové body zprávy o ohlašování směrovače a za jak dlouho je schopen MN provést aktualizaci vazby u HA a u CN.

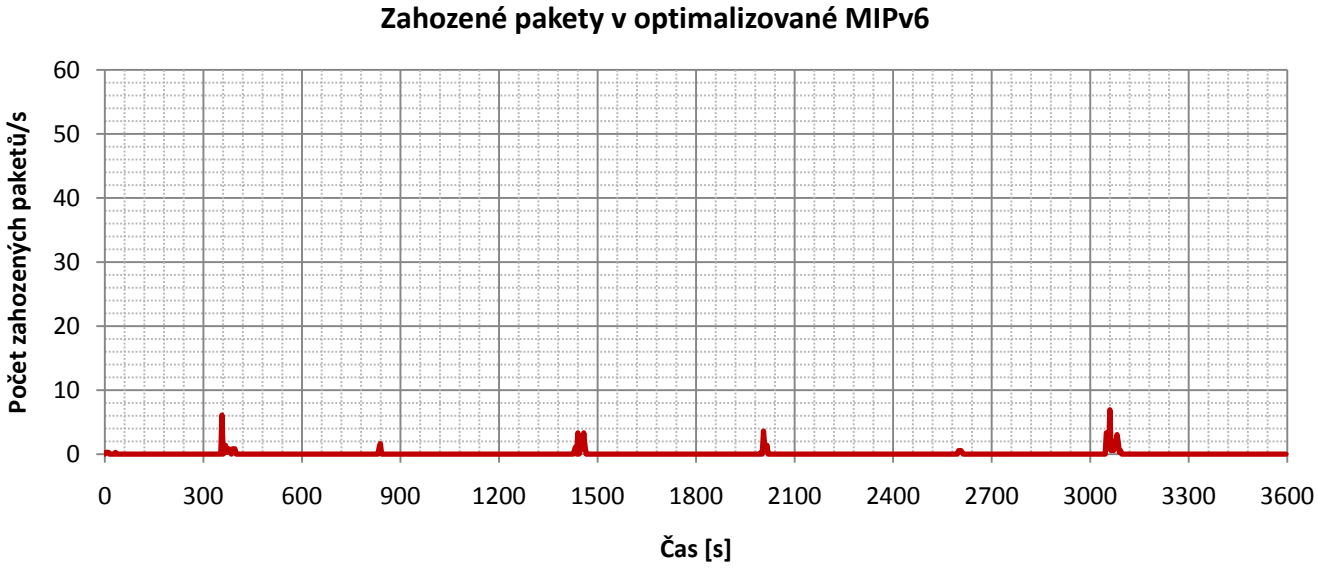
Výsledek 3 (obrázek 7.11)

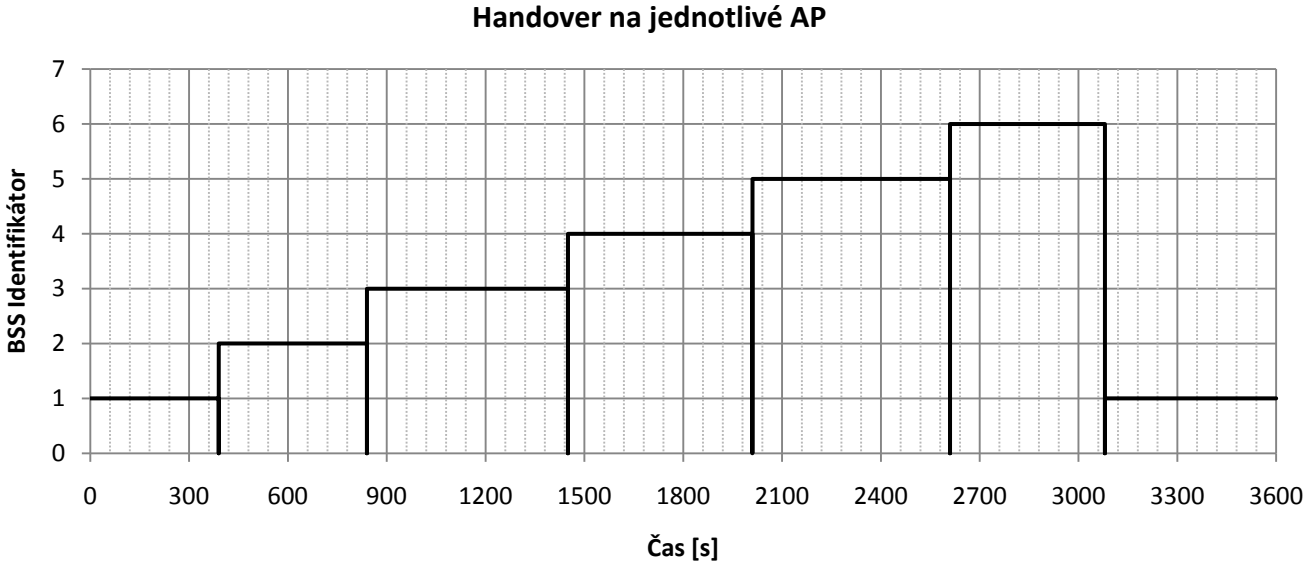
Zde je vidět, že u standardní MIPv6 jsou data v době, kdy je MN mimo domovskou síť, šířena přes HA, kdežto u optimalizované MIPv6 jsou data posílána přímo od CN k MN. Z grafu lze také vyčíst, že optimalizovaná verze MIPv6 je efektivnější, protože má větší propustnost dat. Je to z toho důvodu, že v neoptimalizovaném MIPv6 dochází k opakovanému využívání cest. Pro ukázku si uvedeme příklad komunikace mezi CN a MN v síti FN1. Data, vysílaná od CN budou procházet přes RouterCN, Internet, Router2, HA a k MN půjdou od HA přes Router2, do FN1 a do MN. Cestu mezi HA – Router2 prochází stejná data dvakrát. U optimalizované verze toto nenastane, protože data prochází po provedení RRP mezi MN a CN přímo, ne přes HA.



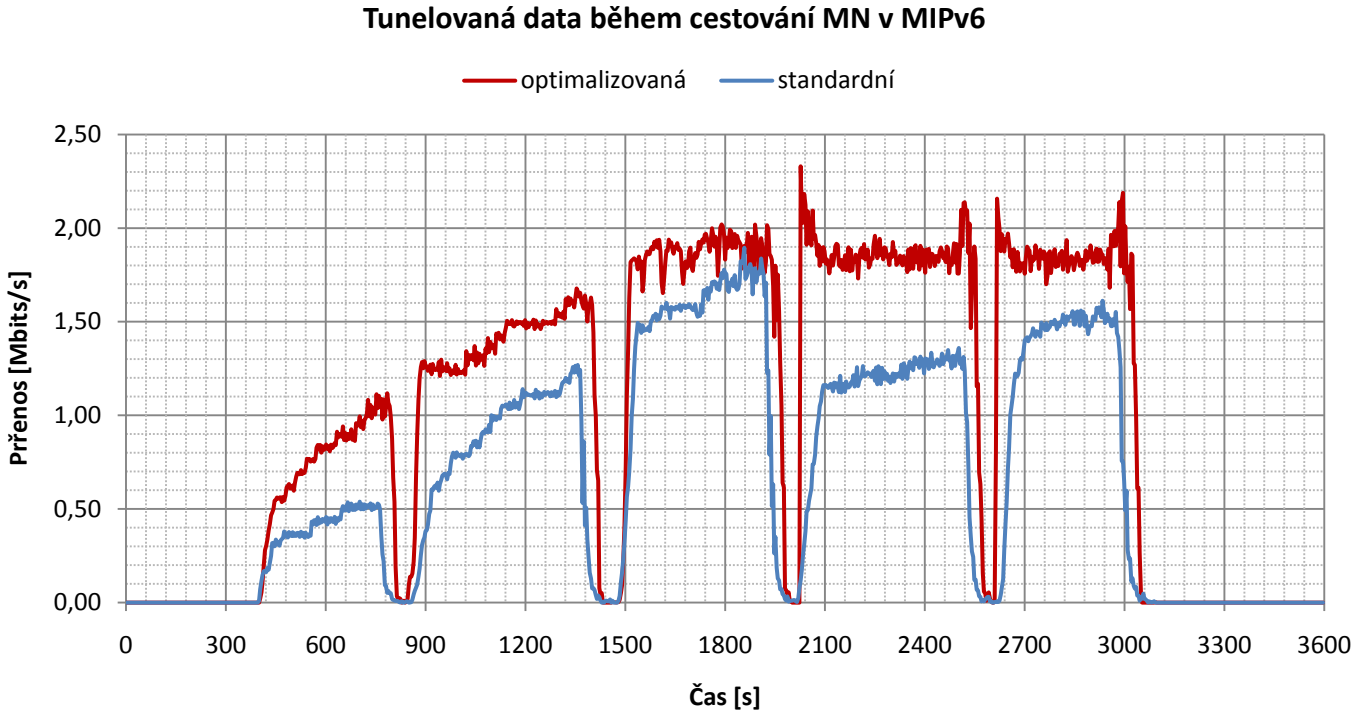
Obr. 7.9: Výsledek1

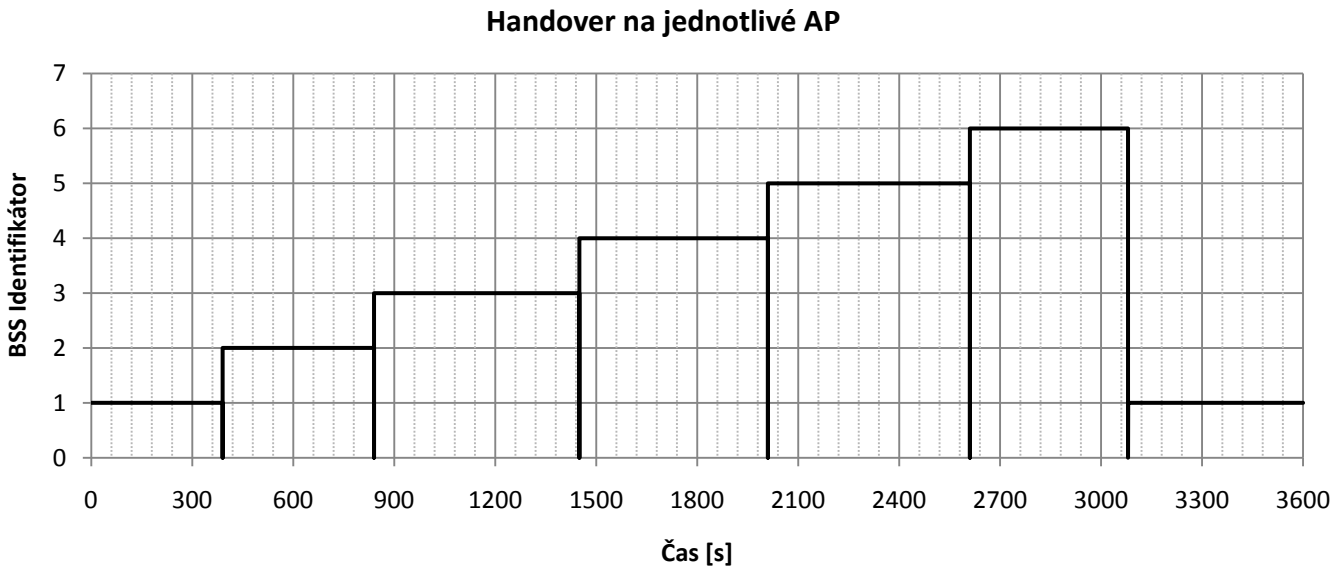






Obr. 7.10: Výsledek2





Obr. 7.11: Výsledek3

8 ANALÝZA ZDROJOVÝCH KÓDŮ MIPV6

Pro zajištění mobility (MIPv6) jsou používány modely procesů *mipv6_mgr* a *mipv6_mn*. Tyto modely jsou potomky *ip* modulu. *Mipv6_mgr* je rodičovským modelem *mipv6_mn*, to znamená, že *mipv6_mn* je volán v případě potřeby z modelu *mipv6_mgr*. V této kapitole se zaměříme na popis konstrukcí a funkcí obou modelů procesů a také bude popsána funkce *wlan_mac*, která má na starosti funkce linkové vrstvy prvků s Wi-Fi rozhranním. Funkce bude popsána pouze s pohledu zajišťování HO, jelikož je popis celkové funkce velmi komplikovaný a obsáhlý. V neposlední řadě se budeme věnovat modelům pro přeposílání paketů v OPNETu, který nám při návrhu FMIPv6 velmi zkomplikuje práci.

8.1 Mipv6_mgr model procesů

V rámci tohoto modelu se provádí čtení atributů důležitých pro zajištění MIPv6, také má na starosti vytváření a vyhodnocování zpráv *BU, BU ack, HoTI, HoT, CoTI, CoT.* V souvislosti s tímto, pracuje model se strukturami *binding cache* a *BU list.* Zajišťuje ještě další funkce, které jsou popsány buďto dále v textu nebo přímo ve zdrojovém kódu procesního modelu.

Uvnitř modelu pracujeme se stavovými proměnnými, které jsou definovány v tabulce 8.1. Následující poznámky se vztahují k tabulkám Tab. 8.1 – Tab. 8.3.

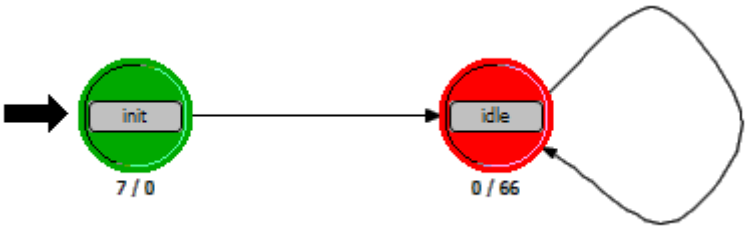
***pozn. 1.:** Tento datový typ je výčtovou množinou prvků, kterých může proměnná tohoto datového typu nabývat. Základní podporované možnosti jsou tyto:

Mipv6C_None	= 0,
Mipv6C_Mobile_Node	= 1,
Mipv6C_Mobile_Router	= 2,
Mipv6C_Home_Agent	= 3,
Mipv6C_Correspondent_Node	= 4,
Mipv6C_Foreign_Router	= 5.

***pozn. 2.:** Tento datový typ je uložen v hlavičkovém souboru *ip_rte_support.h* a obsahuje obsáhlé struktury uchovávající informace pro práci s paketem a uchovává prakticky všechny potřebné informace o aktuálním uzlu v síti (každý uzel má svou strukturu). Tyto informace jsou využívány na síťové vrstvě, konkrétně procesním modelem *ip_dispatch* a je sdílen pro potomky vytvořené IP uzlem (například je tato struktura přístupná i *mipv6_mgr* modelu). V rámci této struktury je tedy zajištěna také podpora mobility a to tím způsobem, že zahrnuje proměnnou *mipv6_info_ptr*, která je popsána v témže hlavičkovém souboru. Do proměnných uložených v *module_data_ptr* lze přistupovat takto: *module_data_ptr->mipv6_info_ptr->(proměnná).* Tabulka 8.2 obsahuje proměnné, do kterých lze tímto způsobem přistupovat.

***pozn. 3.:** Do sdílené paměti se ukládají hodnoty atributů a proměnných. Tato paměť se vytváří na straně rodičovského procesu *mipv6_mgr* a je využívána potomkem *mipv6_mn*.

V rámci modelu *mipv6_mgr* pracujeme s dvěma stavy. Situaci popisuje obrázek 8.1. Vykonávání programu se začíná od stavu *init*. Zde dochází k inicializaci a ke čtení atributů. Ve stavu *idle* se pracuje s rozšiřující hlavičkou pro podporu mobility. Provádí zpracování zpráv MIPv6. Dále si popíšeme části kódu prováděných v jednotlivých stavech, které přímo implementují podporu MIPv6. Ostatní konstrukce nebudou z důvodu velkého rozsahu uvedeny.



Obr. 8.1: Stavový diagram *mipv6_mgr* modelu

Tab. 8.1: Tabulka stavových proměnných *mipv6_mgr* modelu procesů

Datový typ	**Jméno proměnné**	**Komentář**
Mipv6T_Node_Type	mipv6_node_type	ukládá typ uzlu (HA, CN, …) **(*pozn. 1.)**
IpT_Rte_Module_Data*	module_data_ptr	odkaz na data IP **(*pozn. 2.)**
Prohandle	parent_prohandle	odkaz na rodiče procesu
IpT_Ptc_memory*	ip_ptc_mem_ptr	sdílená paměť
Objid	own_mod_objid	ID aktuálního modulu
Objid	own_node_objid	ID aktuálního uzlu
Prohandle	own_prohandle	handler procesu
Miv6T_Mobile_Node_Proc_Shared_Mem*	mobile_host_config_ptr	sdílená paměť pro spolupráci s potomkem procesu **(*pozn. 3.)**
char	proc_model_name[10]	jméno modelu procesů
IpT_Interface_Info*	ha_iface_info_ptr	ukazatel na informace o rozhranní HA
Boolean	Itrace_active	příznak pro označení stavu trasování
Boolean	debug_mode_active	příznak pro označení módu debugeru
char	trace_tittle_str[128]	řetězec používaný pro titulek zpráv debugeru
Mipv6T_Mob_Msg_Proc_Comm*	proc_comm_ptr	paměť pro zprávy MIPv6
double	last_bind_ack_time	čas posledního přijatého potvrzení BU zprávy

Tab. 8.2: Tabulka obsahující proměnné, které zajišťují podporu MIPv6

Datový typ	Jméno proměnné	Komentář
Mipv6T_Node_Type	node_type	význam v tab 8.1
InetT_Address	home_agent_addr	pro uložení IP adresy HA
Boolean	out_of_home	jsem mimo domovskou síť?
Mipv6T_Bind_Cache_Hash_Table*	binding_cache_hashtbl_ptr	odkaz na „cache“ tabulku
Mipv6T_Bind_Update_List_Hash_Table*	binding_update_hashtbl_ptr	odkaz na BU tabulku
Prohandle	mipv6_prohandle	„handle“ mipv6 manažera procesu
Prohandle	mipv6_mn_prohandle	„handle“ mn procesu
InetT_Address*	care_of_addr_ptr	odkaz na aktuální CoA uživatele MN
InetT_Address	home_addr	domovská adresa MN
List*	bind_update_list_lptr	ukládá klíče BU tabulky
List*	bind_cache_table_lptr	ukládá klíče „cache“ tabulky

Kromě struktury pro podporu mobility pro nás mohou být důležité i další, například ty, které uvádí tabulka 8.3.

Tab. 8.3: Další důležité proměnné v procesním modelu mipv6_mgr

Datový typ	Jméno proměnné	Komentář
IpT_Interface_Table	interface_table	Obsahuje informace o všech rozhranních dostupných na daném uzlu.
Objid	module_id	Proměnné typu Objid obsahují identifikátor. Zde se jedná o identifikátory node_id (třeba vlastní ID MN prvku v síti a module_id bude ID ip modulu tohoto prvku.
Objid	node_id	
char*	node_name	Obsahuje jméno uzlu v síti (např. „HA“)

V případě, že nemáme tuto strukturu pro uzel, s k terým chceme pracovat, dostupnou, můžeme si jí zpřístupnit pomocí funkce: ip_support_module_data_get (node_objid). Stačí tedy znát Objid uzlu a dokážeme tuto strukturu vyvolat. Většinou je ale přístupná jako stavová proměnná procesního modelu (*module_data* nebo podobný název).

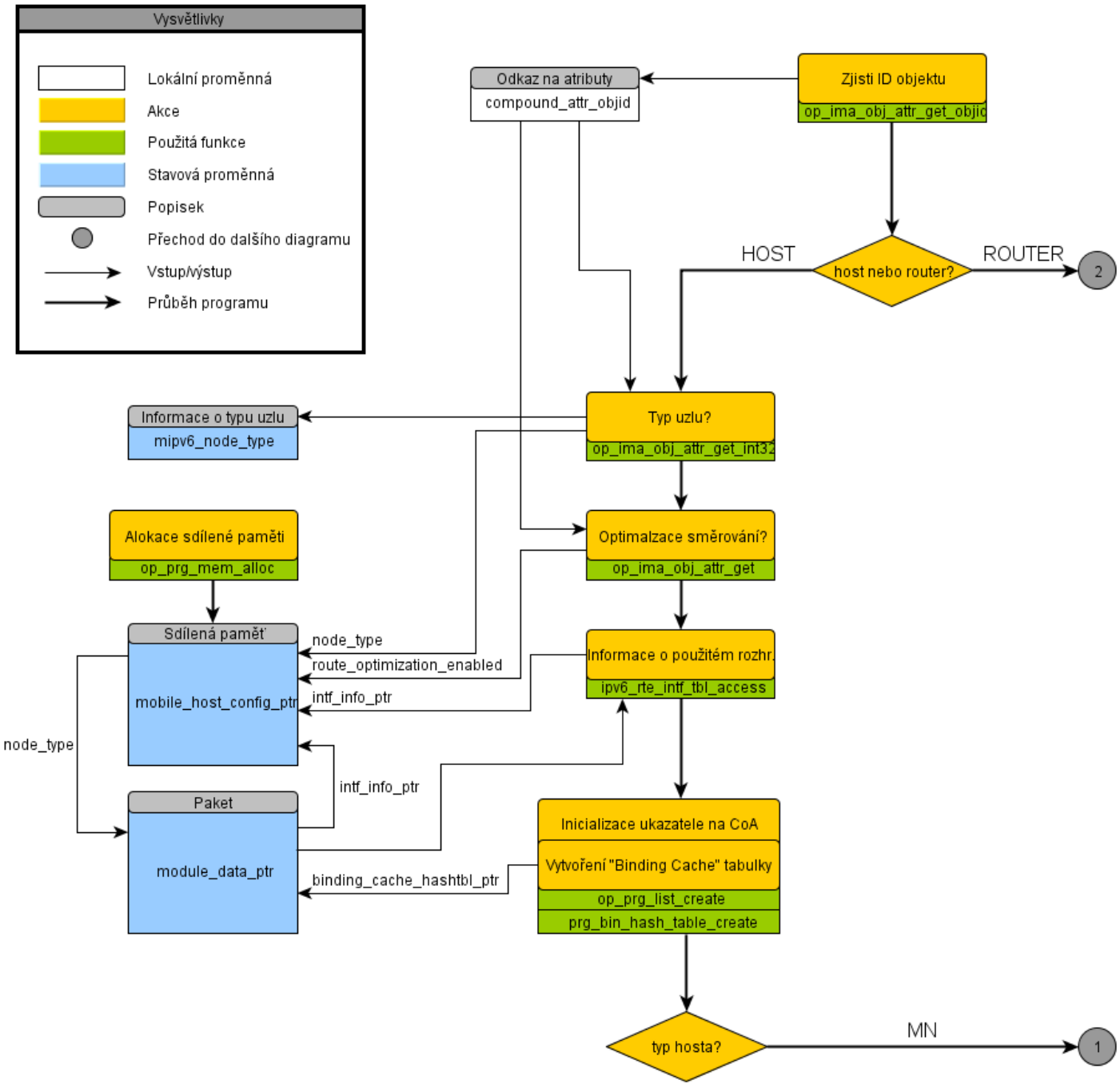
8.1.1 Mipv6_mgr – stav init

Inicializace

V tomto bloku kódu, se provádí registrace modelu procesu a různých souvisejících prostředků pro práci s MIPv6, jako je vyhrazení paměti pro přenos zpráv (použitá proměnná *proc_comm_ptr*). Následuje registrace statistik simulace.

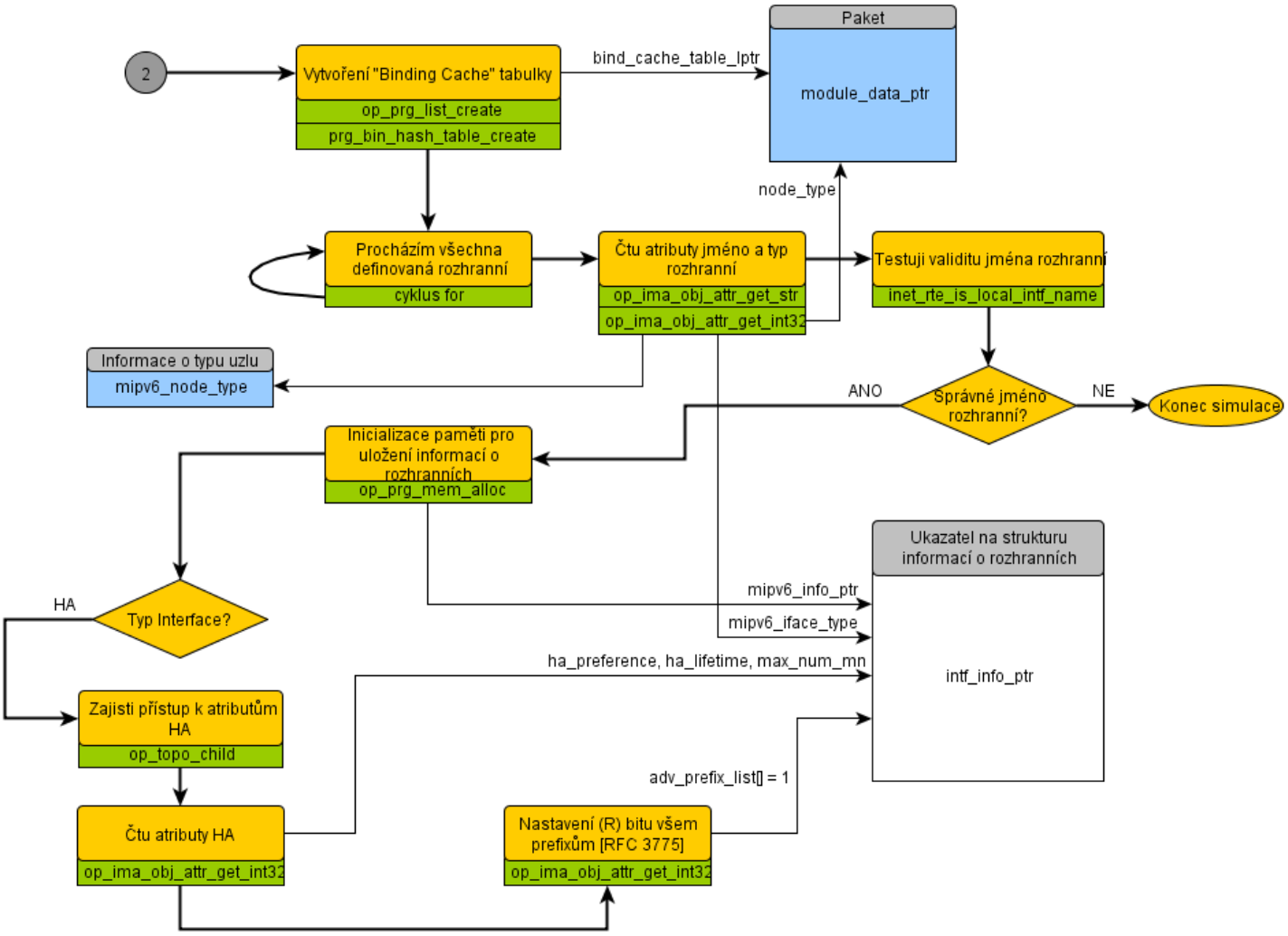
Čtení atributů

Pro čtení atributů se využívá funkce mipv6_attributes_read(). Funkce zpracovává různé atributy, které model potřebuje pro práci s topologicky různými prvky (MN, HA, ...). Funkce ukládá získané atributy do stavových i do svých lokálních proměnných, které jsou definovány uvnitř funkce. Obrázek 8.2 popisuje práci s atributy a proměnnými, s kterými je pracováno na úrovni hosta (uživatele).

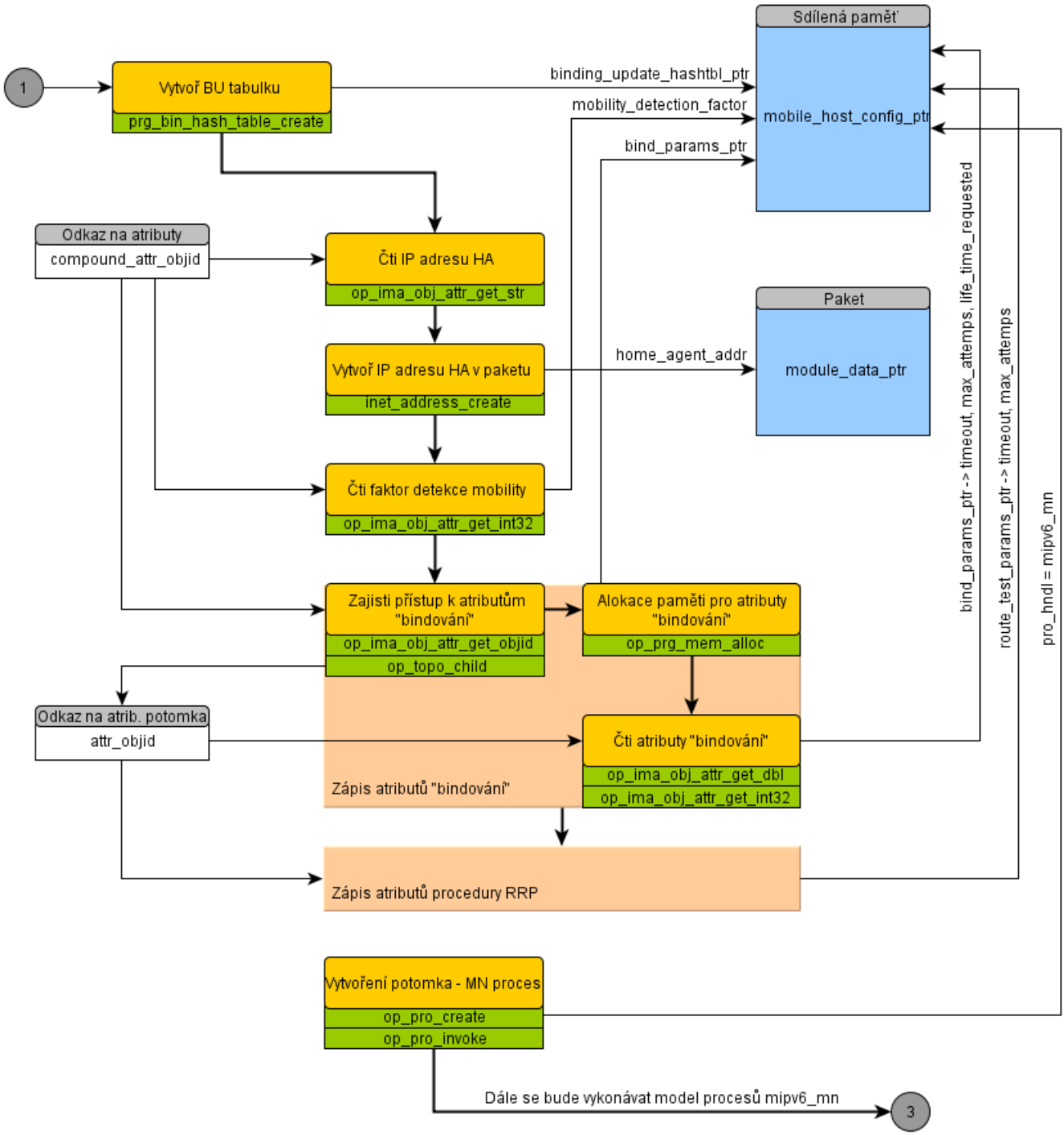


Obr. 8.2: Diagram funkce mipv6_attributes_read – část 1

Na obrázcích 8.3 a 8.4 je zachyceno zpracovávání těchto atributů pro směrovač respektive pro hosta typu MN. Na obrázku 7.3 si všimněme, že na konci diagramu vytváříme vazbu na model *mipv6_mn*. Jedná se o potomka *mipv6_mgr*. Je potřeba mít pod kontrolou, kam se ukládají zjištěné hodnoty z paketu a atributů zadaných pro jednotlivé prvky do simulace, abychom s těmito proměnnými mohli dále pracovat. Důležité jsou pro nás především *compound_atr_objid, mipv6_node_type, mobile_host_config_ptr, module_data_ptr* (popsáno v tab. 7.1). Po provedení načtení atributů následuje stav *idle*.



Obr. 8.3: Diagram funkce *mipv6_attributes_read()* – část 2



Obr. 8.4: Diagram funkce mipv6_attributes_read() – část 3

8.1.2 Mipv6_mgr – stav idle

V prvé řadě je potřeba, aby si tento stav zjistil, kdo ho vyvolal. Je zde naprogramováno nepřímé přerušení a pomocí příkazu invoker_phandle = op_pro_invoker (own_prohandle, &invoke_mode) zjistím, kdo vyvolal toto přerušení. Následuje přístup k paketu pkptr = (Packet*) op_pro_argmem_access (). Z paketu je získána rozšiřující hlavička pro zajištění mobility a zpracována funkcí mipv6_mgr_mobility_msg_process().

Funkce mipv6_mgr_mobility_msg_process()

Tato funkce zpracovává zprávy pro podporu mobility. Ne však všechny. Jedná se o zprávy, které jsou vyřizovány uzlem CN nebo HA. Funkce vyhodnocuje pravidla[8], která je nutné dodržet při zpracování zpráv *HoTI, CoTI, HoT, CoT* a dále řeší operace spojené s přijetím *BU* zprávy. V souvislosti s operacemi se zprávou *BU* probíhají přístupy do struktury *binding cache* na straně HA nebo CN. V případě dodržení pravidel funkce vyvolá mipv6_mgr_mob_msg_send() a prostřednictvím této, odešle potvrzení aktualizace (*BU ack*).

Ostatní zprávy, které inicializuje nebo zpracovává MN, jsou předány potomku, tedy procesu *mipv6_mn*.

Funkce operuje se svými vstupními parametry, které uvádí Tab. 8.3, a také s lokálně definovanými proměnnými, viz. Tab. 8.4. Funkce má také přístup k stavovým proměnným (viz. Tab. 8.1).

Tab. 8.3: Vstupní parametry funkce *mipv6_mgr_mobility_msg_process()*

Datový typ	**Jméno proměnné**
Ipv6T_Mobility_Hdr_Info	*mob_hdr_ptr
IneT_Address	src_addr
IpT_Rte_Ind_Ici_Fields*	intf_ici_fdstruct_ptr

Proměnná *mob_hdr_ptr je ukazatelem na strukturu definovanou v hlavičkovém souboru *ipv6_extension_headers_defs.h* a uchovává informace o přenášeném typu zprávy (v proměnné mh_type). Obsahuje také samotnou přenášenou zprávu (v proměnné msg_data). Následuje proměnná src_addr uchovávající zdrojovou IP adresu. Tedy například při příchodu zprávy *BU* si funkce do této proměnné uloží IP adresu, z které tato zpráva přišla, aby na tuto IP adresu mohla zpětně odpovědět. Datový typ této proměnné je definován v hlavičkovém souboru *ip_addr_v4.h* a umožňuje uložení IPv6 i IPv4 adresy. Posledním vstupním parametrem je intf_ici_fdstruct_ptr*. Přistupuje k informacím o portech.

Tab. 8.4: Některé lokální proměnné funkce *mipv6_mgr_mobility_msg_process()*

Datový typ	**Jméno proměnné**
MIPv6T_Bind_Ack_Status_Value	bind_status
MIPv6T_Bind_Cache_Entry*	bind_entry_ptr
IpT_Interface_Info*	current_ha_iface_info_ptr
IneT_Address	ha_iface_addr

Lokální proměnné dále umožňují uchovávat hodnoty vyčtené ze struktur, ale také uložení hodnot v případě, že proměnná je ukazatelem (končí *..._ptr*). Jsou zde použity datové typy z hlavičkového souboru *mipv6_defs.h*. Jsou to proměnné začínající na *MIPv6T_...*

[8] Pravidla jsou definována v RFC 3775

Pro představu, jak funguje zpracovávání zprávy *BU,* je uveden Obr. 8.5 a 8.6. Ještě uvnitř kódu stavu *Idle* je uložena hlavička mobility do struktury, na kterou ukazuje mob_hdr_ptr. Tato proměnná je definována mezi dočasnými proměnnými (*temporary variable block*) takto:

```
Ipv6T_Mobility_Hdr_Info*    mob_hdr_ptr;
```

Prostřednictvím ukazatele mob_hdr_ptr je přistupováno k hlavičce pro podporu mobility takto:

```
mob_hdr_ptr = (Ipv6T_Mobility_Hdr_Info*) op_prg_list_remove (ipv6_extension_header_list_get (pkt_fields_ptr), OPC_LISTPOS_HEAD);
```

Funkce op_prg_list_remove() odstraní z originální zprávy tuto hlavičku a volání funkce mipv6_mgr_mobility_msg_process() proběhne takto:

```
mipv6_mgr_mobility_msg_process (mob_hdr_ptr, pkt_fields_ptr->src_addr, intf_ici_fdstruct_ptr);
```

Následuje již zpracování *BU* tak, jak je uvedeno na Obr. 8.5. Vysvětlení komplikovanějších konstrukcí kódu:

Získání záznamu o bindování

```
bind_entry_ptr = (Mipv6T_Bind_Cache_Entry *) mipv6_bind_cache_entry_info_get (module_data_ptr, mob_hdr_ptr->msg_data.bind_update.home_address);
```

Do funkce vstupuje ukazatel na paket module_data_ptr (známe tedy jeho obsah) a domovská adresa uživatele MN home_address. Tuto adresu získáme z hlavičky pro podporu mobility tak, že se zanoříme do msg_data, tedy do samotné přenášené zprávy *BU* a z té vyčteme domovskou adresu. Z programátorského hlediska je to řešeno tak, že msg_data je struktura obsahující vnořené struktury zpráv bind_update, bind_ack a bind_error. Tyto jsou definovány v hlavičkovém souboru *mipv6_support.h*. Obsahy jednotlivých zpráv (proměnných) jsou dále definovány v témže hlavičkovém souboru. Ukazatel bind_entry_ptr odkazuje na strukturu datového typu MIPv6T_Bind_Cache_Entry*. Jedná se o odkaz na tabulku obsahující záznamy o „bindování“. Struktura je definována v *mipv6_support.h* takto:

```
typedef struct {
InetT_Address               co_address;
Evhandle                    lifetime_ev;
IpT_Interface_Info *        serving_ha_iface_ptr;
} Mipv6T_Bind_Cache_Entry;
```

Uložení záznamu o bindování

Při splnění podmínek v diagramu na Obr. 8.5 dochází k zapsání CoA, informace o rozhranní obsluhujícího HA a době života „bindování“ do tabulky *binding cache*. Tyto hodnoty jsou získány ze zprávy *BU*. U HA navíc (oproti přístupu u CN) dochází k ověření zdrojů, aby nedošlo k tomu,

že bude obsluhovat více MN, než je mu předepsáno. Samotné volání funkce zajišťující vložení záznamu o „bindování“ do *binding cache* je popsáno takto:

```
bind_entry_ptr = (Mipv6T_Bind_Cache_Entry *) mipv6_bind_cache_entry_info_insert (module_data_ptr, mob_hdr_ptr->msg_data.bind_update.home_address, src_addr, current_ha_iface_info_ptr);
```

Do funkce vstupuje ukazatel na paket module_data_ptr, dále domovská adresa získaná z *BU* zprávy, tedy home_address, potom adresa CoA mobilního uživatele src_addr a nakonec také informace o rozhranní obsluhujícího HA.

Pozn.: Existuje-li již záznam v *binding chache* se stejnou CoA, nedojde k aktualizaci tohoto záznamu, pokud je to vyžádáno, bude zasláno pouze potvrzení *BU ack*.

Vykonávání programu může postupovat také dle Obr. 8.6, pakliže *lifetime* ve zprávě *BU* bude nulové nebo menší. V tomto případě odesílatel *BU* zprávy žádá o vymazání svého záznamu z *binding cache*.

Odstranění záznamu o bindování

Odstranění záznamu z *binding cache* dosáhneme voláním následující funkce takto:

```
bind_entry_ptr = (Mipv6T_Bind_Cache_Entry*) mipv6_bind_cache_entry_info_remove (module_data_ptr, mob_hdr_ptr->msg_data.bind_update.home_address);
```

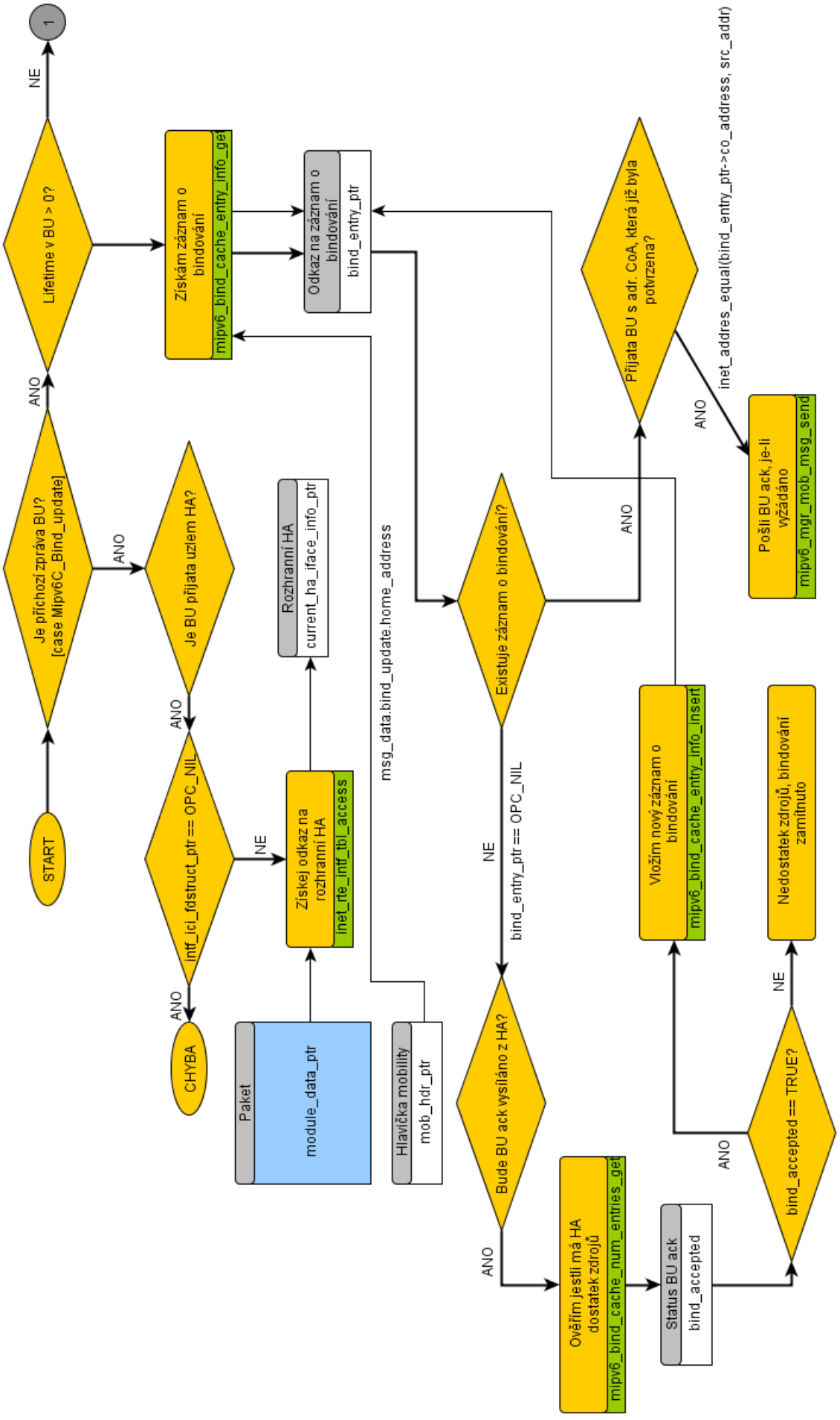
Odstraním záznam v paketu, na který se odkazuji pomocí module_data_ptr pro adresu uloženou v home_address v hlavičce pro podporu mobility ve zprávě *BU*.

Odeslání potvrzení o provedené aktualizaci vazby

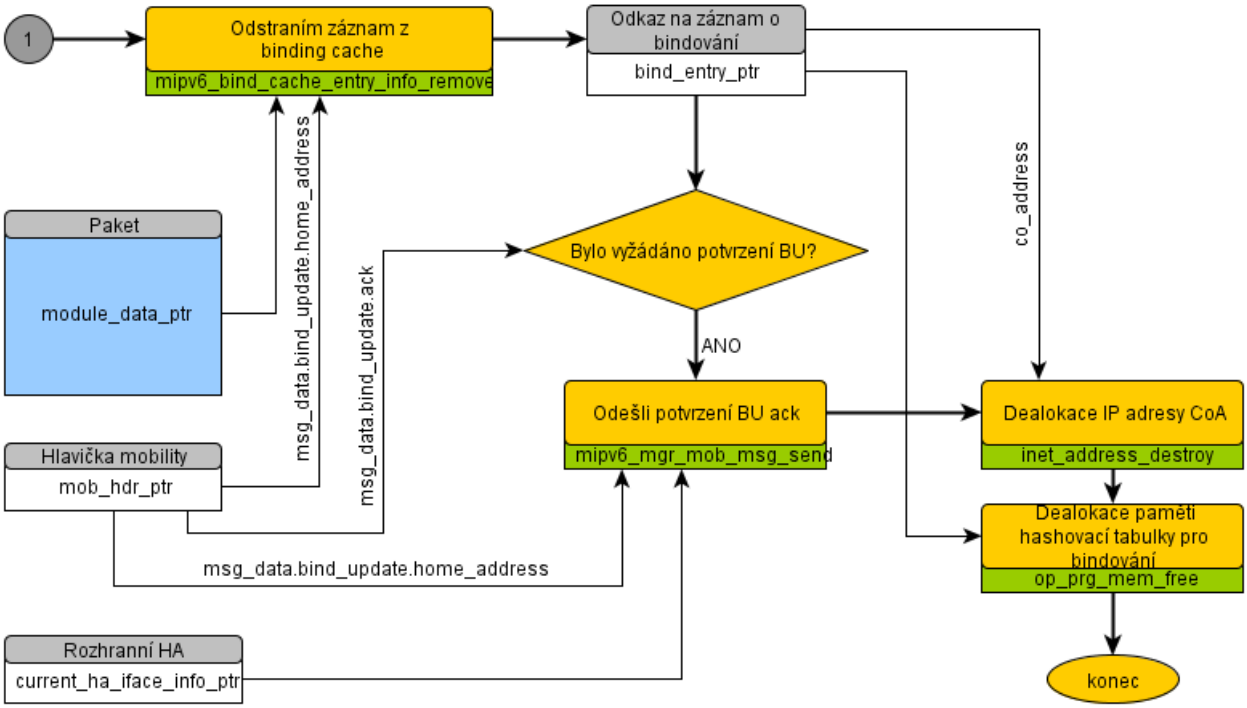
Odeslání zprávy *BU ack* je zajištěno funkcí mipv6_mgr_mob_msg_send(). Tato funkce je také popsána ve zdrojovém kódu k procesnímu modelu *mipv6_mgr*. Funkce vytvoří a odešle IPv6 datagram, který nese *BU ack* zprávu. Zpráva je volána takto:

```
mipv6_mgr_mob_msg_send (Mipv6C_Bind_Ack, current_ha_iface_info_ptr, mob_hdr_ptr -> msg_data.bind_update.home_address, Mipv6C_Binding_Update_Accepted, 0);
```

Vstupními parametry funkce jsou typ přenášené zprávy, zde je to zpráva *BU ack*, dále sem vstupují informace o rozhranní obsluhujícího HA, potom domovská adresa MN, informace o stavu „bindování“ a o době života.



Obr. 8.5: Diagram zprávy BU, tak jak je zpracována funkcí *mipv6_mgr_mob_msg_send()* – část1



Obr. 8.6: Diagram zprávy BU, tak jak je zpracována funkcí *mipv6_mgr_mob_msg_send()* – část2

Funkce mipv6_mgr_mob_msg_send()

Jak již bylo popsáno výše, funkce se stará o odesílání zprávy *BU* ack. Funkce je definována takto:

mipv6_mgr_mob_msg_send (Mipv6T_Mobity_Hdr_Type msg_type, IpT_Interface_Info* current_ha_iface_info_ptr, InetT_Address dest_addr, Mipv6T_Bind_Ack_Status_Value status, int lifetime)

Tedy vstupují do ní informace o přenášeném typu zprávy msg_type, o rozhranní obsluhujícího HA current_ha_iface_info_ptr, dále adresa, na kterou bude potvrzení odesláno dest_addr, potom status *BU* status a nakonec doba života lifetime. Kromě těchto vstupních parametrů, využívá funkce ještě stavové proměnné a své lokální proměnné.

Průběh vykonávání kódu této funkce je takový, že nejprve je vytvořen IP diagram pomocí ip_pktr = ip_dgram_create() a je vytvořena jeho struktura pomocí ip_dgram_fd_ptr = ip_dgram_fdstruct_create (). Následuje naplnění struktury datagramu údaji. Jedná se o adresy odesílatele src_addr a příjemce dest_addr zprávy[9]. Údaje získáme ze sdílené paměti mobile_host_config_ptr v případě, že paket bude sestavovat MN nebo CN. V případě, že paket bude sestavovat HA, budou údaje získány z current_ha_iface_info_ptr. Údaje se vkládají například takto:

ip_dgram_fd_ptr->src_addr = inet_rte_intf_addr_get (mobile_host_config_ptr->intf_info_ptr, InetC_Addr_Family_v6);

[9] Jsou zapisovány ještě další údaje. Lze je vyčíst ze zdrojového kódu funkce.

Posledním úkolem této funkce, je vytvořit ukazatel na rozšiřující hlavičku, připojit ji k IP diagramu a zaslat. Takže při vytváření zprávy *BU ack* postupuji takto:

```
Switch (msg_type)
{
    case Mipv6C_Bind_Ack:
    mob_hdr_ptr = (Ipv6T_Mobility_Hdr_Info *) mipv6_binding_ack_msg_create (status, 0, (OpT_uInt16)
lifetime);
break;
}
ipv6_mobility_hdr_insert (ip_dgram_fd_ptr, mob_hdr_ptr);
ip_dgram_fields_set (ip_pkptr, ip_dgram_fd_ptr);
op_pk_nfd_access (ip_pkptr, "fields", &ip_dgram_fd_ptr);
    ip_dgram_sup_ipv6_extension_hdr_size_add (&ip_pkptr, &ip_dgram_fd_ptr,
IpC_Procotol_Mobility_Ext_Hdr, (int) ext_hdr_len);
mipv6_mgr_ip_pkt_send_delayed (ip_pkptr, (msg_delivery_time - current_sim_time));
```

Postup zde popsaný je využit při návrhu zpráv pro implementaci FMIPv6.

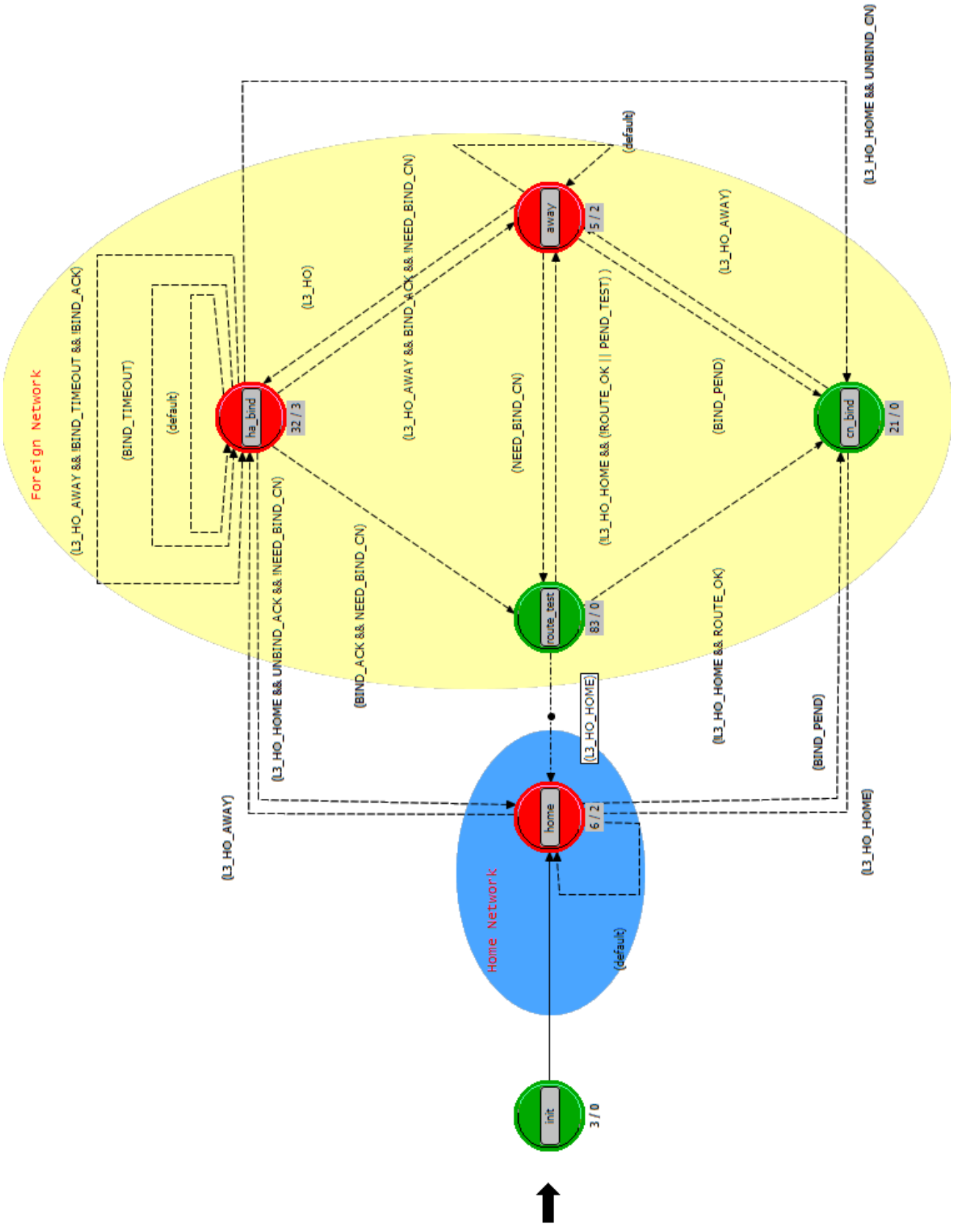
8.2 Mipv6_mn model procesů

Zpracování probíhá na základě stavového diagramu uvedeného na Obr. 8.7. V tomto procesním modelu dochází k zpracování různých stavů pro MN, ke kterým dochází při cestování tohoto uživatele sítěmi. Chování tohoto modelu lze popsat pomocí tabulky událostí, viz. Tab. 8.5.

Tab. 8.5: Tabulka podmínek přechodů mezi stavy pro mipv6_mn

Aktuální stav	**Podmínka**	**Nový stav**
init	-	home
home	L3_HO_AWAY	ha_bind
	BIND_PEND	cn_bind
	default	home
ha_bind	L3_HO_HOME && UNBIND_ACK && !NEED_BIND_CN	home
	BIND_ACK && NEED_BIND_CN	route_test
	L3_HO_AWAY && BIND_ACK && !NEED_BIND_CN	away
	L3_HO_HOME && UNBIND_CN	cn_bind
	L3_HO_AWAY && !BIND_TIMEOUT && !BIND_ACK	ha_bind
	BIND_TIMEOUT	
	default	
away	NEED_BIND_CN	route_test
	BIND_PEND	cn_bind
	L3_HO	ha_bind
	default	away
cn_bind	L3_HO_HOME	home
	L3_HO_AWAY	away

route_test	L3_HO_HOME	home
	!L3_HO_HOME && ROUTE_OK	cn_bind
	!L3_HO_HOME && (!ROUTE_OK || PEND_TEST)	away



Obr. 8.7: Stavový diagram procesního modelu mipv6_mn

Podmínky logických proměnných nabývají hodnoty 1 = *TRUE* nebo 0 = *FALSE*. Jejich významy jsou následující:

L3_HO_AWAY = Mobilní uživatel je mimo svojí domovskou síť. Vyhodnocení probíhá díky funkci mipv6_layer3_handoff_process(). Tato proměnná je přednastavena na hodnotu FALSE.

L3_HO_HOME = Mobilní uživatel je v domovské síti. Tato proměnná je přednastavena na hodnotu TRUE.

BIND_PEND = Tento příznak v případě že je pravdivý, říká mobilnímu uživateli, že *return routability procedure* proběhla úspěšně (MN přijal zprávu CoT) a nyní korespondent čeká na aktualizaci vazby. Proměnná je přednastavena na hodnotu FALSE.

UNBIND_ACK = Tento příznak v případě že je pravdivý, říká mobilnímu uživateli, že se úspěšně odhlásili od domácího agenta. Situace nastane v případě, že MN nevyužívá služeb HA – tedy pakliže je MN v domovské síti. Proměnná je přednastavena na hodnotu FALSE.

NEED_BIND_CN = V případě že je tento příznak pravdivý, bude se spouštět *return routability procedure*. MN je v cizí síti a dosud neprovedl „bindování" na CN. Proměnná je přednastavena na hodnotu FALSE.

BIND_ACK = Tento příznak v případě že je pravdivý, říká mobilnímu uživateli, že HA žádosti o „bindování" vyhověl. Je užitečné například proto, že chci-li spustit *return routability procedure* musím již tou dobou být přístupný přes HA. Proměnná je přednastavena na hodnotu FALSE.

UNBIND_CN = Příznak ohlašující potřebu odregistrovat MN z *binding cache* v CN. Tento příznak je však podivně využit. Je nastaven na hodnotu 0 a při běhu programu se jeho hodnota nemění. Ve stavu *cn_bind* regulérně dochází k vymazání *binding cache* při návrtu MN do domovské sítě, ale díky použití nesplnitelné podmínky mezi stavy *ha_bind* a *cn_bind*, k přechodu mezi těmito stavy nikdy nedojde. Proměnná je přednastavena na hodnotu FALSE.

BIND_TIMEOUT = Tento příznak v případě že je pravdivý, říká mobilnímu uživateli, že vypršel čas potřebný pro přijetí *BU ach* na straně MN. Proměnná je přednastavena na hodnotu FALSE.

L3_HO = Tento příznak bude pravdivý v případě, že se MN bude pohybovat z domovské sítě do cizí sítě nebo mezi dvěma cizími sítěmi. Hodnotu této proměnné mění funkce mipv6_layer3_handoff_process().Proměnná je přednastavena na hodnotu FALSE.

ROUTE_OK = Tento příznak má v celém procesním modelu hodnotu FALSE. Nastává podobně nesmyslně navržený přechod mezi stavy *route_test* a *cn_bind*, ke kterému nikdy nedojde. Ze stavu *route_test* mohu přejít buďto do stavu *home* nebo do stavu *away*.

PEND_TEST = Příznak je použit pro označení stavu *return routability procedure*. Nabývá-li hodnoty TRUE, jsem ve stavu *route_test* a právě probíhá vykonávání této procedury. Nabývá-li hodnoty FALSE, jsem ve stavu *away*.

8.2.1 Dostupné datové struktury a proměnné

Procesní model *mipv6_mn* vzniká jako potomek *mipv6_mgr*. Díky sdílené paměti mezi těmito procesními modely, je možné, aby *mipv6_mn* operoval s hodnotami, které jsou uloženy na straně *mipv6_mgr*. Sdílená paměť je reprezentována strukturou mobile_host_config_ptr. Struktura je postupně naplňována různými hodnotami tak, jak je vidět na vývojových diagramech na obrázcích 8.2 – 8.4. Na straně modelu *mipv6_mn* je struktura přístupná skrze stavovou proměnnou mn_config_mem_ptr. Další strukturu, která si uchovává informace pro práci s paketem je module_data_ptr. Proměnné, které obsahuje, jsou uvedeny v Tab. 8.2. Také je přenášena struktura proc_comm_ptr, která uchovává odkaz na hlavičku pro podporu mobility a zdrojovou IP adresu zprávy MIPv6, pro potřebu zpracování zprávy *mipv6_mn* procesním modelem.

8.2.2 Popisy jednotlivých stavů a vybraných funkcí

Stav *home*

Ve stavu *home* je MN v domovské síti. V tomto stavu může započít nebo ukončit provádění MIPv6. Z tohoto stavu se může dostat do *ha_bind* v případě, že MN vycestuje nebo se může dostat do *cn_bind*, pakliže se MN vrací do *home.*

Stav *cn_bind*

Uvnitř tohoto stavu se provádí „bindování" MN na CN nebo se toto „bindování" ruší. Aktualizace vazby je provedena funkcí mipv6_mn_bind_update_msg_send() s dobou života 1000. Vymazání záznamu z *binding cache* CN je provedeno zasláním stejné funkce, ale s dobou života 0.

Stav *ha_bind*

V tomto stavu MN posílá zprávu *BU* domácímu agentu. Mohou zde nastat dvě situace. MN je v domácí síti, pošle tedy zprávu *BU* s dobou života 0 nebo je MN v cizí síti a pošle tedy *BU* s dobou života 1000. Na základě nastavených příznaků pak funkce mipv6_mn_invokation_process() určí, kam bude směřovat další vykonávání programu.

Stav away

V tomto stavu se vykonává program, je-li MN vycestovaný mimo svojí domovskou síť.

Funkce mipv6_mn_bind_update_msg_send()

mipv6_mn_bind_update_msg_send (InetT_Address dest_addr, Boolean ack_requested, Boolean home_reg, OpT_uInt16 lifetime)

Funkce vytváří zprávu *BU* a je zasílána na adresu, kterou určuje proměnná dest_addr mezi vstupními proměnnými této funkce. Zasílána je na adresu HA nebo CN uživatelem MN.

Vytvářením zprávy je myšleno vytvoření paketu pomocí ip_pkptr = ip_dgram_create (), vytvoření struktury paketu pomocí ip_dgram_fd_ptr = ip_dgram_fdstruct_create () a naplnění této struktury (zdrojová a cílová adresa, typ přenášeného protokolu, délku polí, atd.). K tomuto standardnímu IPv6 datagramu připojím hlavičku pro podporu mobility s obsahem, který odpovídá přenášenému typu zprávy. Vytvořím ji takto:

```
mob_hdr_ptr = (Ipv6T_Mobility_Hdr_Info *)mipv6_binding_update_msg_create (module_data_ptr -> mipv6_info_ptr->home_addr, 0, ack_requested, home_reg, lifetime);
```

a připojím takto:

```
ipv6_mobility_hdr_insert (ip_dgram_fd_ptr, mob_hdr_ptr);
```

Vytvořenou strukturu budu přenášet jako paket, který mám definován pomocí stavové proměnné. Přiřazení provedu takto:

```
ip_dgram_fields_set (ip_pkptr, ip_dgram_fd_ptr);
```

Následuje ještě aktualizace polí paketu, přizpůsobení délky a pak již můžu paket odeslat, takto:

```
mipv6_mn_ip_pkt_send (ip_pkptr);
```

Funkce mipv6_mn_invokation_process()

Funkce reaguje na jednotlivé zprávy MIPv6, které do ní mohou vstoupit díky nepřímému přerušení od *mipv6_mgr* modelu. Nepřímé přerušení vznikne při předání vykonávání programu od rodiče k potomku [14]. Při nepřímém přerušení zde dojde k předání zprávy a funkce nastavuje hodnoty pravdivosti logických proměnných. Tyto logické proměnné potom budou sloužit pro řízení přechodů mezi různými stavy uvnitř modelu *mipv6_mn*. Funkce také umožňuje řízení pomocí přímého přerušení, a sice umožňuje přerušení vlastní (*self*) i od jiného procesu (*process*). Tabulka 8.4 shrnuje možnosti zpracování přerušení touto funkcí.

Tab. 8.4: Obsluha přerušení v *mipv6_mn_invokation_process()*

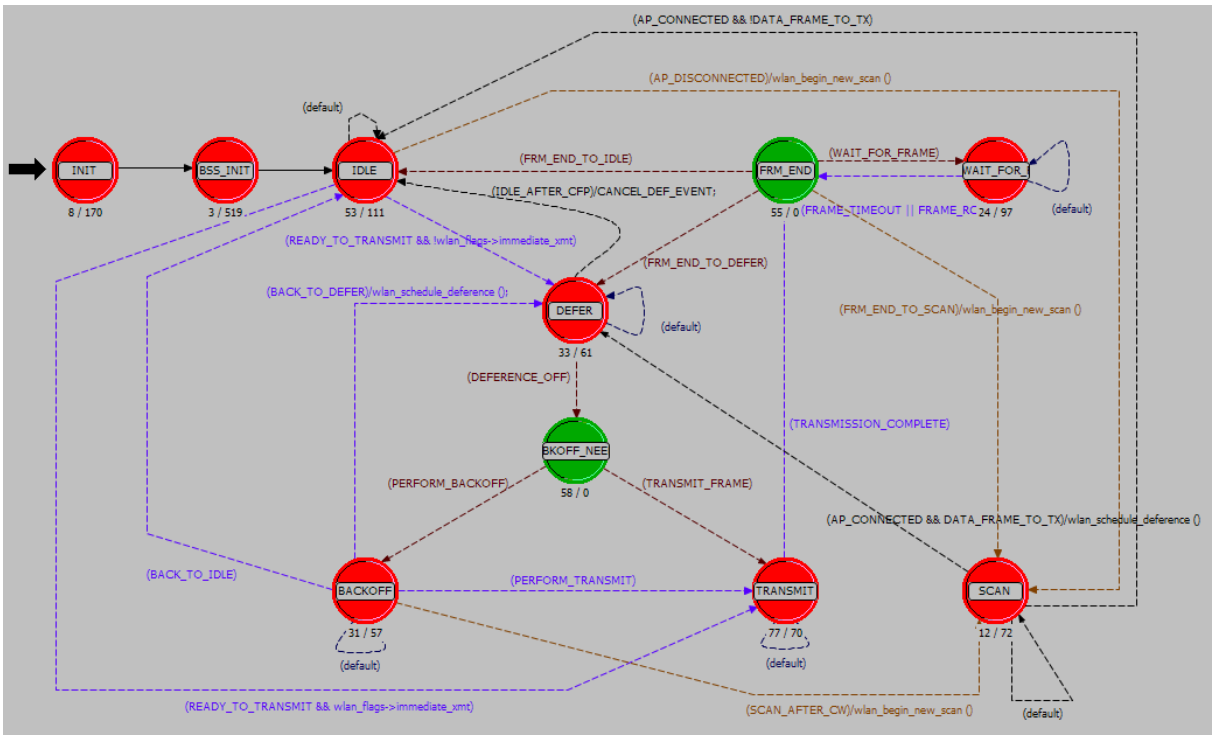
Pojmenování reakce	**Typ reakce**	**Typ přerušení**
Mipv6C_Bind_Ack	zpráva od *mipv6_mgr*	nepřímé
Mipv6C_Home_Test	zpráva od *mipv6_mgr*	nepřímé
Mipv6C_Care_Of_Test	zpráva od *mipv6_mgr*	nepřímé
Mipv6C_Bind_Error	zpráva od *mipv6_mgr*	nepřímé
Mipv6C_Bind_Ref_Req	zpráva od *mipv6_mgr*	nepřímé
IPC_RA_MOBILITY_DETECTION_INTRPT_CODE	od procesu	přímé
MIPV6C_START_BINDING_INTRPT_CODE	od procesu	přímé
BINDING_UPDATE_RETX_TIMER_CODE	vlastní	přímé
ROUTE_TEST_RETX_TIMER_CODE	vlastní	přímé

Funkce mipv6_layer3_handoff_process()

Je funkce, která zpracovává „handover“ na síťové vrstvě. Funkce je spuštěna, zjistí-li MN, že se vyskytl v cizí síti. Toto zjištění může vzniknout přijetím „router advertisement“ z cizí sítě. Pro nás není studium této funkce až tak důležité, protože FMIPv6 není spouštěna pokynem ze síťové vrstvy, ale z linkové.

8.3 Wlan_mac

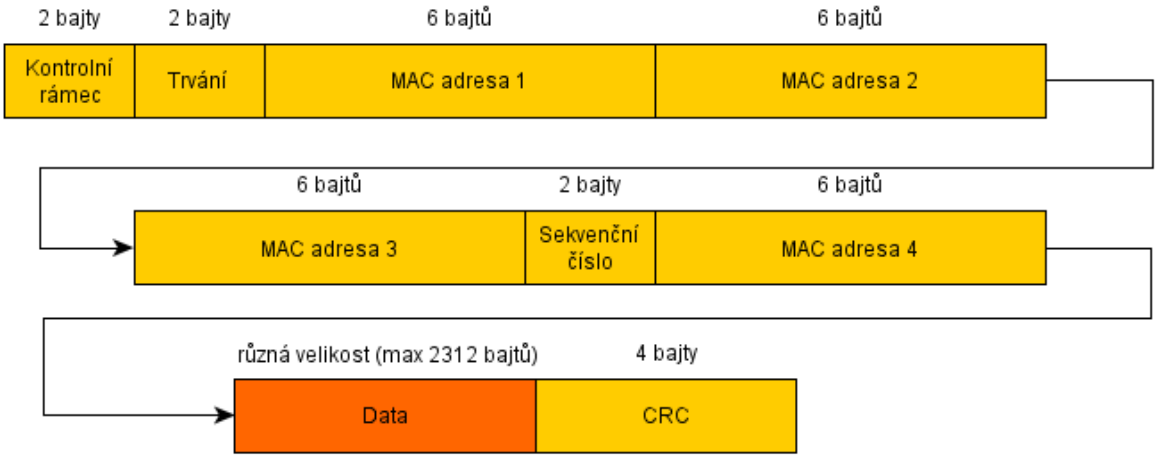
Tento modul zajišťuje funkce spojené s přístupem k bezdrátové síti na druhé vrstvě (linkové). Procesní model je zobrazen na obr. 8.8. Pro nás je důležitá funkce spojená s dostupností aktuální AP. Tuto informaci musíme pro návrh FMIPv6 znát, protože řídí rozhodnutí o průběhu tohoto protokolu.



Obr. 8.8: Procesní model *wlan_mac*

Dostupnost aktuálního AP je vyhodnocována ve funkcích wlan_ap_reliability_eval() a wlan_ap_eval_virtual(). Tyto funkce jsou k nalezení ve zdrojovém kódu v souboru *wlan_support.ex.c* v adresáři std/wireless_lan OPNETu. Obě funkce jsou vyhodnocovány periodicky, z informací pocházejících z šíření rámců „beacon“ od AP (wlan_ap_reliability_eval()), případně na základě polohy MN v síti (wlan_ap_eval_virtual()). Jedná se tedy o pasivní metodu určení dostupnosti AP. Informace pocházející z aktivního scanování (ve stavu *scan*) na obr. 8.8 nás v momentu spouštění FMIPv6 nezajímají, protože v momentě scanování je již MN odpojen od AP, a tudíž již nemůže být spuštěn prediktivní režim FMIPv6, který bude v této práci implementován.

Kromě informace o aktuální velikosti signálu AP a událostech s ní spojenými je nutné přenést z linkové do síťové vrstvy také MAC adresu prku NAR. Informaci přeneseme prostřednictvím ICI (jelikož je nar_mac datového typu OpT_Int64, použijeme pro přiřazení hodnoty do ICI op_ici_attr_set_int64()). Konkrétní řešení lze nalézt v kapitole 9.1.1. Nejprve si však musíme říci, jak je možné tuto informaci získat. Bezdrátová architektura Wi-Fi přenáší několik různých typů rámců, mají však něco společné [17]. A sice to, že přenáší informace o MAC adresách (vysílací a přijímací AP, odesílatele a příjemce). Rámec zobrazuje Obr. 8.9.



Obr. 8.9: Wi-fi rámec

MAC adresa 1 je adresa rádiového přijímače, MAC adresa 2 je adresa rádiového vysílače, MAC adresa 3 je adresa cíle (příjemce rámce) a MAC adresa 4 je adresa zdroje (odesílatele rámce). Nás zajímá MAC adresa 4 a bude vyčtena rámce „Beacon“.

8.4 Modely pro přeposílání paketů

Alespoň základní znalost těchto modelů je pro nás důležitá, protože je nutné přizpůsobit návrh hlaviček zpráv FMIPv6 mechanizmům zpracovávající pakety na úrovni síťové vrstvy. A je také potřeba přizpůsobit již implementované funkce pro přijímání těchto nových zpráv (paketů).

OPNET poskytuje několik mechanismů pro směrování (přeposílání) paketů. Jsou to "Slot Based" (v OPNETu se jedná o procesní model *ip_rte_slot*) a "Central Processing" (*ip_rte_central_cpu*). Takzvané "cloudy" používají odlišný model (*ip_rte_cloud*). Existuje ještě model *ip_rte_distrib_cpu*. Tyto procesní modely jsou volány rodičovským ip_dispatch [18].

Při odesílání paketu se mohou stát dvě věci: Paket chceme poslat do vyšší vrstvy, tedy v rámci jednoho uzlu nebo chceme paket poslat do jiného uzlu v síti. Směrovače v naší simulaci (MIPv6 i v návrhu FMIPv6) využívají model *ip_rte_central_cpu*, proto následující analýza funkce přeposílání paketů bude provedena pro tento model.

8.4.1 ip_rte_central_cpu model

Tento procesní model má na starosti příjem i odesílání paketů. Je volán z procesního modelu *ip_dispatch*. Obsahuje funkce pro příjem paketů (ip_rte_packet_arrival()) a pro odeslání paketu (ip_rte_packet_send()). Funkce ip_rte_packet_send() je definována v hlavičkovém souboru *ip_rte_support.h* takto:

ip_rte_packet_send (IpT_Rte_Module_Data* module_data, Packet* pkptr, Ici* pk_ici_ptr, IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr, IpT_Rte_Process_Type process_type, void* process_info_ptr);

První dva parametry již byly vysvětleny dříve (jedná se o data uzlu a paket). Další parametry jsou pk_ici_ptr, což je ukazatel na ICI přidružené k paketu a intf_ici_fdstruct_ptr, což je ukazatel na strutkturu ICI obsahující informace o rozhranních (jedná se o pole "rte_info_fields" ICI struktury přidružené k paketu a obsahuje informace důležité pro směrování paketu. Process_type je v tomto modelu nastaven na IpC_Rte_Process_Type_Central_Cpu a poslední parametr je nastaven na OPC_NIL.

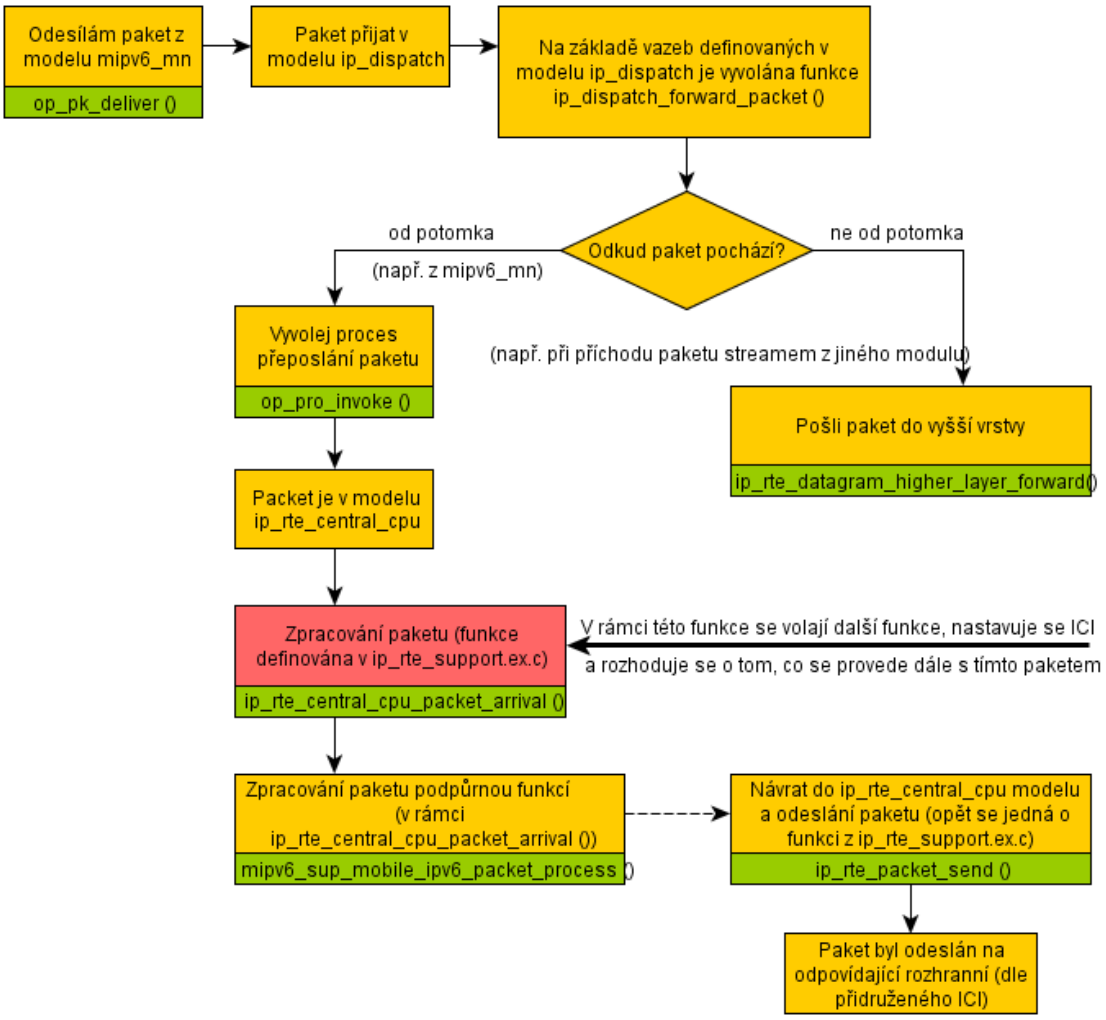
Záznam ICI přidružený k paketu je naplňován ve funkci ip_rte_packet_arrival(). Tato funkce je volána z modelu *ip_rte_central_cpu*, je-li potřeba zpracovat příchozí paket. Jedná se o obsáhlou funkci. Ip_rte_packet_arrival() zpracovává odlišně pakety přicházející z vyšších vrstev a pakety přicházející z nižších vrstev ISO. Tato funkce se stará o:

- Kontrolu IP diagramu,
- spouštění navazujících funkcí při vyhodnocení různých typů paketů (například při vyhodnocování paketu na straně zařízení podporující mobilitu, je příchozí paket zpracováván ještě funkcí mipv6_sup_mobile_ipv6_packet_process()),
- zamítnutí nežádoucího paketu v případě že je přijat firewallem a tento paket je nežádoucí (funkce se vyhodnocuje pouze pro uzel firewallu),
- vytvoření a naplnění struktury ICI + přiřazení této struktury k paketu (op_ici_create() ("ip_rte_ind_v4") ← ip_rte_ind_ici_fdstruct_create(); op_pk_ici_set()). Struktura je naplňována dle dat v module_data vyhodnocujícího uzlu nebo dle informací přicházejících s paketem. Datová struktura IpT_Rte_Ind_Ici_Fields je definována hlavičkovém souboru *ip_dgram_sup.h*.

Pro nově implementované hlavičky paketů musíme zajistit podporu na straně ip_rte_arrival() funkce. Potažmo tedy musíme zajistit podporu u funkcí zpracovávající takovýto IPv6 paket. Jelikož funkce ip_rte_packet_arrival() zachází s pakety relativně transparentně, stačí upravit funkci mipv6_sup_mobile_ipv6_packet_process() (popsáno v kapitole 9.2.4). Funkce je definována takto:

mipv6_sup_mobile_ipv6_packet_process (IpT_Rte_Module_Data* iprmd_ptr, IpT_Dgram_Fields** pk_fd_pptr, Packet **pkpptr, Boolean packet_from_lower_layer)

Do funkce vstupují data z uzlu v síti (iprmd_ptr), struktura IPv6 datagramu (pk_fd_pptr), samotný paket (pkpptr) a příznak informující o paketu z nižší vrstvy. Tato funkce se stará o zpracování paketů MIPv6. Zpracování spočívá především v práci s rozšiřujícími hlavičkami, které dle potřeby protokolu MIPv6 do paketů přidává. Zpracování je odlišné pro různé typy uzlů sítě. Funkce je k nalzení v knihovně OPNETu v *mipv6_sup.ex.c*. Model *ip_rte_central_cpu* odesílá paket podle obrázku 8.10. Paket je odeslán z procesního modelu *mipv6_mn* a v *ip_dispatch* modelu (rodičovský) je obsloužen funkcí ip_dispatch _forward_packet(). Paket je stále v uzlu MN, dokud není odeslán na odpovídající rozhranní definované v přidruženém ICI k paketu. Před samotným odesláním je tedy ještě v rámci *ip_rte_central_cpu* modelu volána funkce pro zpracování příchozího paketu a poté je paket odeslán. Při přijímání paketů (např. na straně AP) je opět vyvolána funkce ip_rte_packet_arrival(), nyní se však vykoná jiný kód než při odesílání. Pakety nyní budou vyhodnocovány jako pakety pocházející z nižší vrstvy.



Obr. 8.10: Schéma odesílání paketu z mipv6_mn

Bohužel lze velmi těžko pochopit, proč při zpracovávání paketů s podporou mobility musí vykonávání programu v OPNETu procestovat tolik různých podpůrných knihoven. Jednotlivé typy rozšiřujících hlaviček jsou zpracovávány na různých místech (i v rámci síťové vrstvy) a celé řešení je roztrženo do několika podpůrných zdrojových knihoven (*mipv6_sup.ex*, *ipv6_extension_headers_sup.ex.c*, část signalizace MIPv6 ještě zpracovává *mipv6_signaling _sup.ex.c*) a vše je pochybně propojeno.

Mipv6_sup.ex.c obsahuje funkce pro vytváření procesního modelu *mipv6_mgr*, pro zpracovávání IPv6 paketů s podporou mobility a pro zápis do statistik (přijetí a odesílání řízení provozu MIPv6, data přijatá a odeslaná prostřednictvím tunelu a další).

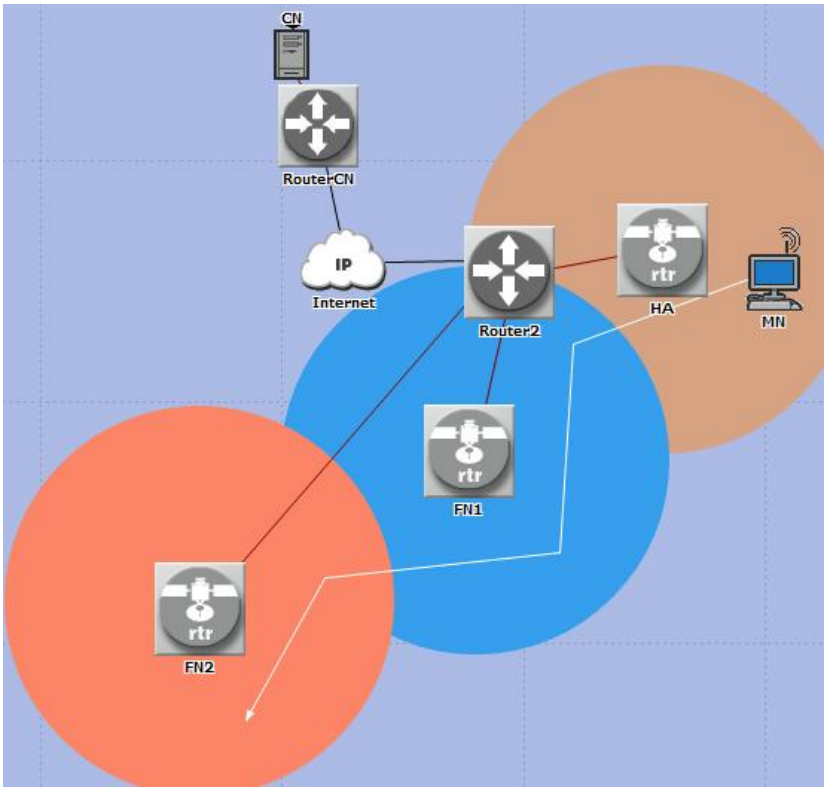
Ipv6_extension_headers_sup.ex.c obsahuje funkce pro vytváření a zpracovávání rozšiřujících hlaviček. Jsou zde k nalzení funkce pro práci s IPv6 rozšiřujícími hlavičkami „směrování (typ2)“, „mobility“ a „destination“ („destination“ - bude použita zkratka DH). Například při zpracovávání hlaviček s podporou mobility je zde zajištěno, aby se vykonávání programu předalo do *mipv6_mgr* procesního modelu (pomocí op_pro_invoke()).

Mipv6_signaling_sup.ex.c obsahuje funkce pro vytváření zpráv pro podporu mobility i funkce pro práci se strukturami „binding cache“ nebo „binding update list“. Také jsou zde funkce pro zabalování a vybalování IPv6 paketů do resp. z tunelu (je potřeba při přenášení MIPv6 paketů mezi HA a MN). Většina těchto funkcí je voláno z procesních modelů *mipv6_mgr* nebo *mipv6_mn*.

9 IMPLEMENTACE FMIPV6

V této kapitole bude popsána implementace protokolu FMIPv6 do stávajícího modelu MIPv6. Implementaci předcházela podrobná analýza funkčních mechanismů protokolu MIPv6 a konstrukčních mechanismů programu OPNET. Výsledky rešerše jsou shrnuty v kapitole 8. Implementace bude provedena na modelu simulace, který je uveden na Obr. 9.1.

V následujícím textu je uvedena implementace spouštění FMIPv6 protokolu z linkové vrstvy. Je zde také popsáno řešení šíření těchto událostí z linkové vrstvy na síťovou vrstvu MN. Dále jsou implementovány zprávy RtSolPr a PrRtAdv protokolu FMIPv6. Těmito zprávami přenáším mezi MN a AP potřebné informace o handoveru do následující cizí sítě. Jelikož jsou tyto zprávy v OPNETu nové, bylo nutno jim zajistit podporu mezi několika procesními modely. Zprávy jsou přenášeny rozšiřující hlavičkou „destination“ (bude použita zkratka DH). Jelikož přidání podpory pro tyto zprávy zabralo mnoho času, nebylo v daném času možné implementovat celý protokol FMIPv6. Nicméně řešení níže uvedené je funkční.



Obr. 9.1: Model simulace pro FMIPv6

9.1 Detekce handoveru a výměna informací mezi L2 a L3

Z *wireless_lan_mac* potřebujeme získat informace o poklesu signálu z aktuálně připojeného AP a také MAC adresu NARu (prvku v síti, ke kterému se bude MN připojovat). Tyto informace získáme z funkcí popsaných v kapitole 8.3.

9.1.1 Implementace událostí na linkové vrstvě

Úprava spočívá v přidání kódu do funkce wlan_ap_eval_virtual() uvnitř *wlan_support.ex.c.*

```
/*    Jestlize se zmensi prijimany vykon pod urven WLANC_ROAM_SCAN_START_FMIPV6_THRESH a budou splneny dalsi podminky, indikuj zapnuti FMIPV6. Pozn.: detekci spusteni chci provest pouze jednou pro udalost, proto je zde jeste vnoreno rozhodnuti o aktualnim nastaveni      */

if (  (ap_rx_pow < WLANC_ROAM_SCAN_START_FMIPV6_THRESH(rx_pow_threshold))          &&
      (sta_roam_info_ptr->fmipv6_L2_start == OPC_FALSE)                           &&
      (sta_roam_info_ptr->par_odpojen == OPC_TRUE))

{
      sta_roam_info_ptr->par_odpojen = OPC_FALSE;
      sta_roam_info_ptr->fmipv6_L2_start = OPC_TRUE;
```

```
}
```

a do *wlan_ap_reliability_eval()* v témže zdroji.

```
/*    Jestlize spadne dostupnost AP pod WLANC_AP_FMIPV6_START_THRESHOLD, nastavim priznak pro spusteni FMIPv6                */

if (  (roam_state_ptr->ap_reliability < WLANC_AP_FMIPV6_START_THRESHOLD)                    &&
      (roam_state_ptr->fmipv6_L2_start == OPC_FALSE)                                       &&
      (roam_state_ptr->par_odpojen == OPC_TRUE))

{
      roam_state_ptr->par_odpojen = OPC_FALSE;
      roam_state_ptr->fmipv6_L2_start = OPC_TRUE;

 }
```

fmipv6_L2_start, par_odpojen je logická proměnná informující o rozhodnutí k provedení FMIPv6. Proměnné, ještě spolu s nar_pripojen jsou doplněny do struktury informací o roamingu (do roam_state_ptr v hlavičkovém souboru *wlan_support.h*). Rozhodnutí se řídí pomocí definovaných konstant WLANC_ROAM_SCAN_START_FMIPV6_THRESH a WLANC_AP_FMIPV6_START _THRESHOLD. Konstanty jsou definovány ve vlastním vytvořeném hlavičkovém souboru (*fmipv6_support.h*), který je připojený zdrojovému kódu pomocí direktivy *include*.

Rozhodnutí o připojení k následujícímu AP (NAR) je řešeno ve stavu *SCAN* v procesním modelu *wlan_mac* takto:

```
if (eval_bss_id != bss_id)
{
      wlan_ap_switch ();
      //prave probehlo prepnuti na novou AP
      //nastavim tedy priznaky pro roaming tak, abych byl schopen opet spustit FMIPv6
      roam_state_ptr->par_odpojen = OPC_TRUE;
      roam_state_ptr->nar_pripojen = OPC_TRUE;
      roam_state_ptr->fmipv6_L2_start = OPC_FALSE;
}
```

Logické proměnné řídí reakci na tuto událost v modelu *wlan_mac*.

Jelikož FMIPv6 vyhodnocuje HO na základě informací poskytnutých druhou vrstvou, bylo nutné implementovat komunikaci mezi druhou a třetí vrstvou. Tedy mezi moduly *wireless_lan_mac* a *ip*. Kromě samotné detekce aktuální hodnoty signálu AP, je v kódu také vyřešen přenos MAC adresy následujícího AP. Implementace je provedena ve stavu *IDLE*(viz. obr. 8.8) v modelu *wlan_dispatch* následovně:

```
/*    POZNAMKY K IMPLEMENTACI:

Rozhodnuti o spusteni FMIPv6 je provedeno na zaklade dostupnosti soucasneho AP
[wlan_ap_reliability_eval] a hodnote prijimaneho signalu od soucasneho AP, ktery je odvozen od
vzdalenosti [wlan_ap_eval_virtual].        */

// zrusim cekajici udalosti, ktere byly vyvolany pri minule detekci a nyni jiz nejsou platne
if (op_ev_valid (preruseni_evh))
    op_ev_cancel (preruseni_evh);

//je-li nastaven priznak fmipv6_L2_start nebo nar_pripojen na TRUE , spoustim pripravu k volani do IP
if ((roam_state_ptr->fmipv6_L2_start == OPC_TRUE) || (roam_state_ptr->nar_pripojen == OPC_TRUE))
{
    /*  je potreba objevit registraci tohoto procesu pomoci oms_pr_process_discover (hledam dle
    "process name" a "protocol"). Pozn.: Na druhe strane je potreba provest registraci mipv6_mn
    modelu pomoci oms_pr_process_register    */

    //vytvoreni listu pro potrebu ulozeni
    list_objevenych_registraci_ptr = op_prg_list_create ();
    oms_pr_process_discover(OPC_OBJID_INVALID, list_objevenych_registraci_ptr,
            "process name",     OMSC_PR_STRING,     "mipv6_mn",
            "protocol",         OMSC_PR_STRING,     "mipv6",
            OPC_NIL);

    //ziskam zaznam z listu
    handle_procesu = (OmsT_Pr_Handle) op_prg_list_remove (list_objevenych_registraci_ptr,
        OPC_LISTPOS_HEAD);

    //list jiz nepotrebuji, smazu ho
    op_prg_mem_free (list_objevenych_registraci_ptr);

    //pro provedeni preruseni do mipv6_mn potrebuji ziskat objid modulu IP.
    oms_pr_attr_get (handle_procesu, "id_modul_ip", OMSC_PR_OBJID, &muj_ip_objid);

    //vytvorim ICI a ulozim do nej data, ktera budu sirit prerusenim
    ici_do_ip = op_ici_create("info_z_wlan_mac");

    //(1) podminka, ktera vyvola preruseni s kodem pro spusteni fmipv6
    if ((roam_state_ptr->fmipv6_L2_start == OPC_TRUE)        &&
        (roam_state_ptr->par_odpojen == OPC_FALSE))
    {
        nar_bss_id = bss_id + 1;
        nar_mac = wlan_get_ap_sta_addr(nar_bss_id);
        //ICI
        op_ici_attr_set (ici_do_ip, "bss_id", bss_id);
        op_ici_attr_set_int64 (ici_do_ip, "nar_mac", nar_mac);
        op_ici_attr_set (ici_do_ip, "detekce_fmip", roam_state_ptr->fmipv6_L2_start);
        op_ici_attr_set (ici_do_ip, "par_odpojen", roam_state_ptr->par_odpojen);
```

```
            op_ici_attr_set (ici_do_ip, "nar_pripojen", roam_state_ptr->nar_pripojen);
            op_ici_install(ici_do_ip);

            preruseni_evh = op_intrpt_schedule_remote (op_sim_time(), IPC_FMIPV6_INTRPT_CODE,
            muj_ip_objid);

            //po provedeni preruseni vratim puvodni hodnoty
            roam_state_ptr->fmipv6_L2_start = OPC_FALSE;
            roam_state_ptr->par_odpojen = OPC_FALSE;
        }

        //(2) detekce odpojeni od PARu – není implementovano

        //(3) detekce pripojeni k NARu - odeslani odpovidajiciho přerušeni
        if ((roam_state_ptr->nar_pripojen == OPC_TRUE)          &&
            (roam_state_ptr->par_odpojen == OPC_TRUE)           &&
            (roam_state_ptr->fmipv6_L2_start == OPC_FALSE))
        {
            //ICI
            op_ici_attr_set (ici_do_ip, "bss_id", bss_id);
            op_ici_attr_set (ici_do_ip, "detekce_fmip", roam_state_ptr->fmipv6_L2_start);
            op_ici_attr_set (ici_do_ip, "par_odpojen", roam_state_ptr->par_odpojen);
            op_ici_attr_set (ici_do_ip, "nar_pripojen", roam_state_ptr->nar_pripojen);
            op_ici_install(ici_do_ip);

            preruseni_evh = op_intrpt_schedule_remote (op_sim_time(), IPC_NAR_INTRPT_CODE,
                    muj_ip_objid);

            //po provedeni preruseni vratim puvodni hodnoty
            roam_state_ptr->fmipv6_L2_start = OPC_FALSE;
            roam_state_ptr->par_odpojen = OPC_TRUE;
            roam_state_ptr->nar_pripojen = OPC_FALSE;
        }

}
```

Řešením je tedy princip objevení registrace *mipv6_mn*, získání Objid na *ip* modul a volání pomocí vzdáleného přerušení do *ip* modulu. Tímto přerušením následně přenášíme kód (IPC_FMIPV6_INTRPT_CODE, nebo IPC_PAR_INTRPT_CODE, nebo IPC_NAR_INTRPT_CODE) a informaci o bss_id aktuálně připojeného AP a také MAC adresu následujícího AP do modulu *ip*. Kód přerušení slouží pro rozlišení události u příjemce přerušení a informace o bss_id aktuálního AP je šířena prostřednictvím ICI. Postup vytvoření ICI je popsán v kapitole 6.2.1. Pozor! Při rozšiřování sítě o další AP je NUTNÉ dodržet, aby jednotlivé přístupové body, jejichž sítěmi bude MN procházet, měli vzestupné číslování BSS ID, rostoucí o jedna ve směru pohybu MN. Je to z důvodu, aby byla správně vyčíslena MAC adresa následujícího AP. V simulaci je přiřazení BSS ID následující (Tab. 9.1):

Tab. 9.1: Přiřazení čísla BSS ID prvkům v simulaci FMIPv6

Název AP v simulaci	HA	FN1	FN2
BSS IS	1	2	3

V kódu jsou použity logické proměnné par_odpojen, nar_pripojen a fmipv6_L2_start. Tyto proměnné řídí spouštění FMIPv6 z modelu *wlan_mac*. Bylo nutné uvážit jejich volbu, aby bylo řízení přerušení do modulu *ip* provedeno regulárně ve spolupráci s detekcí událostí zavedených ve *wlan_support.ex.c* a ve stavu *SCAN* modelu *wlan_mac*. Hodnoty těchto logických proměnných, které vyvolávají přerušení do modulu *ip* uvádí následující tabulka, Tab. 9.2. Dalším přínosem je, že díky tomuto řešení bude detekce FMIPv6 fungovat i v síti s větším počtem AP, ke kterým se bude MN připojovat.

Tab. 9.2: Zvolené hodnoty pro řízení událostí FMIPv6 z linkové vrstvy

Logická proměnná	**Výchozí hodnoty**	**Detekce FMIPv6**	**Připojení k NARu**
fmipv6_L2_start	OPC_FALSE	OPC_TRUE	OPC_FALSE
par_odpojen	OPC_TRUE	OPC_FALSE	OPC_TRUE
nar_pripojen	OPC_FALSE	-	OPC_TRUE

9.1.2 Detekce přerušení na síťové vrstvě

Detekce přerušení na straně *ip* modulu je řešena v procesním modelu *ip_dispatch* ve stavu *IDLE.exit* následujícím způsobem:

```
/*    Jestlize nastane preruseni od wlan_mac, predej informaci procesnimu modelu mipv6_mn. Spolu s
prerusenim posli i informace z L2. MN si na zaklade techto informaci bude ridit protokol FMIPv6.    */

// zrusim cekajici preruseni, byla jiz zpracovana
if (op_ev_valid (preruseni_evh))
      op_ev_cancel (preruseni_evh);

if (  (intrpt_code == IPC_FMIPV6_INTRPT_CODE)              | |
         (intrpt_code == IPC_PAR_INTRPT_CODE)              | |
         (intrpt_code == IPC_NAR_INTRPT_CODE))

{
         // diagnostika v ODB
         preruseni = OPC_TRUE;
         cas_preruseni_v_ip = op_sim_time();

         // vem ici
         muj_ici_ptr = op_intrpt_ici();
         op_ici_attr_get (muj_ici_ptr, "bss_id", &bss_id);
         op_ici_attr_get_int64 (muj_ici_ptr, "nar_mac", &nar_mac);
         op_ici_attr_get (muj_ici_ptr, "detekce_fmip", &fmip_detekovano);
```

```
        op_ici_attr_get (muj_ici_ptr, "par_odpojen", &par_odpojen);
        op_ici_attr_get (muj_ici_ptr, "nar_pripojen", &nar_pripojen);

        // preposli ho dal
        op_ici_install(muj_ici_ptr);

        // proved preruseni na mipv6_mn - prohandle tohoto procesu je ulozen v
        // module_data.mipv6_info_ptr->mipv6_mn_prohandle
        mipv6_mn_phndl = module_data.mipv6_info_ptr->mipv6_mn_prohandle;

        // dle detekovaneho typu provedu take odpovidajici preruseni do mipv6_mn
        if (intrpt_code == IPC_FMIPV6_INTRPT_CODE)
        {
                preruseni_evh = op_intrpt_schedule_process (mipv6_mn_phndl, op_sim_time(),
                        IPC_FMIPV6_INTRPT_CODE);
        }
        else if (intrpt_code == IPC_NAR_INTRPT_CODE)
        {
                preruseni_evh = op_intrpt_schedule_process (mipv6_mn_phndl, op_sim_time(),
                        IPC_NAR_INTRPT_CODE);
        }

}
```

Do *ip_dispatch* byly přeneseny požadované informace a nyní je potřeba je dále šířit k *mipv6_mn* modelu, který dál bude s těmito informacemi pracovat. Zde již může být použito přerušení pomocí op_intrpt_schedule_process(), protože modul *ip* zná „prohandle“ na procesní model, kterému je rodičem. Tento „prohandle“ je uložen ve struktuře module_data.mipv6_info_ptr ->mipv6_mn_prohandle. V modelu *mipv6_mn* je přerušení standardně vyzvednuto ve funkci mipv6_mn_invokation_process(). K předchozímu textu o této funkci (tato funkce je popsána v kapitole 8.2.2) je nutno dodat, že nyní kromě nepřímého přerušení bude zpracovávat také naše přímé přerušení od modelu *ip_dispatch*. Implementace vypadá v této funkci následovně (řádek 585 v kódu):

```
intrpt_type = op_intrpt_type ();
intrpt_code = op_intrpt_code ();

if (intrpt_code == IPC_FMIPV6_INTRPT_CODE)
{
        preruseni_z_wlan_mac = OPC_TRUE;

        // nactu potrebne informace z ICI
        muj_ici_ptr = op_intrpt_ici();
        op_ici_attr_get (muj_ici_ptr, "bss_id", &bss_id);
        op_ici_attr_get_int64 (muj_ici_ptr, "nar_mac", &nar_mac);
```

```
        // ICI jiz nepotrebuji, protoze jsem si info o bss id a nar_mac jiz ulozil
        op_ici_destroy (muj_ici_ptr);

        // akce na prichozi preruseni od wlan_mac
        // spust fmip, je-li to opravdu zadouci - podminky uvnitr funkce
        if (preruseni_z_wlan_mac)
              fmipv6_mn_rtsolpr_msg ();
        //neni-li to zadouci vykonavani programu z funkce vyskoci, nebude vytvoren paket
        //nastav preruseni z wlan_mac na FALSE
        preruseni_z_wlan_mac = OPC_FALSE;

// k preruseni timto kodem dojde vzdy, kdyz MN detekuje, ze signal ze soucasne AP je pod urovni
        // spoustejici FMIPv6
}
```

9.2 Implementace zpráv RtSolPr a PrRtAdv

V následujícím textu bude popsána implementace zpráv RtSolPr a PrRtAdv. Zprávy budou navrženy jako IPv6 datagram s rozšiřující hlavičkou „destination options“. OPNET má implementovanou podporu pro rozšiřující hlavičku „routing“ (typ 2), „destination options“ a „mobility“. Při návrhu bylo zohledněno zpracování zpráv funkcemi definovanými v *ipv6_extension_headers_sup.h* a dalšími, jako například ip_rte_packet_arrival(), ip_rte_packet_send(), atd. Celá komunikační cesta mezi MN-PAR (obecně AP) musela být modifikována tak, aby nedocházelo k problémům při přenosu nového typu zpráv. Problémy, které tento fakt přináší, jsou rozebrány v kapitole 8.4.1. nebo dále v textu. V případě, že bylo potřeba modifikovat zdrojové kódy, je postup naznačen v následujících kapitolách.

9.2.1 Funkce pro odeslání RtSolPr zprávy

Vykonávání protokolu FMIPv6 začíná ve stavu *AWAY* z procesního modelu *mipv6_mn* uzlu MN (viz. Obr. 8.7), pakliže přijde přerušení z *wlan_mac* (řádek 591 *mipv6_mn* v bloku funkcí: if (intrpt_code == IPC_FMIPV6_INTRPT_CODE)), detekující úbytek signálu na současné AP jsou uloženy informace obsažené v ICI a je spuštěna funkce fmipv6_mn_rtsolpr_msg(), která vyšle paket s hlavičkou pro cíl (DH) s obsahem zprávy RtSolPr dle Obr. 3.2. Funkce je definována v hlavičkovém souboru *fmipv6_support.h* a je implementována v bloku funkcí v procesním modelu *mipv6_mn* následovně:

```
//funkce pro odeslani rtsolpr zpravy, vcetne podminek nutnych pro odeslani
static void fmipv6_mn_rtsolpr_msg (void)
{
        //funkce vytvori paket a posle ho aktualnimmu AP - PARu.
        /* Podminky: Prislo preruseni z wlan_mac? Jsem venku? Jsem dostupny pres HA? Mam CoA? */
        Packet*                                 ip_packet_ptr;
        IpT_Dgram_Fields*                       ip_dgram_fd_ptr;
```

```
        //struktura IP datagramu - jeho datovy typ je definovan v ip_dgram_sup.h
        OpT_Packet_Size                         delka_datagramu;        //v bitech
        Ipv6T_Destination_Hdr_Info*             destination_hdr_ptr;    //vlastni datovy typ
            // rozsirujici hlavicky (viz ipv6_extension_headers_defs.h)

FIN (fmipv6_mn_rtsolpr_msg (void));

if (  (preruseni_z_wlan_mac == OPC_TRUE)            &&
      (L3_HO_AWAY)                                  &&
      (!NEED_BIND_CN)                               &&
      (!inet_address_equal (  (module_data_ptr->mipv6_info_ptr->home_agent_addr), (module_data_ptr
            ->mipv6_info_ptr->care_of_addr_ptr)  )  )
)
{
                        /*      =====TVORBA IPV6 DATAGRAMU======        */
      //delka datagramu vcetne rozsirujici hlavicky
      delka_datagramu = 128;

      //vytvoreni paketu
      ip_packet_ptr = ip_dgram_create ();

      //paklize nebude mozne vytvorit paket => chyba => konec
      If (ip_packet_ptr == OPC_NIL)
            op_sim_end ("Nebylo mozne vytvorit IP datagram.", "","", "");

      //vytvorim strukturu odpovidajici zakladnimu IP datagramu a inicializuji jeho pole
      ip_dgram_fd_ptr = ip_dgram_fdstruct_create ();

      //naplnim datagram hodnotami
      //zdrojova adresa je CoA
      ip_dgram_fd_ptr->src_addr           = inet_address_copy (*(module_data_ptr->mipv6_info_ptr->
            care_of_addr_ptr));
      ip_dgram_fd_ptr->src_internal_addr  = inet_rtab_addr_convert (ip_dgram_fd_ptr->src_addr);

      //cilova adresa je adresa PARu
      ip_dgram_fd_ptr->dest_addr          = inet_address_copy (default_router_addr);
      ip_dgram_fd_ptr->dest_internal_addr = inet_rtab_addr_convert (ip_dgram_fd_ptr->dest_addr);

      //delka datagramu bude
      ip_dgram_fd_ptr->orig_len           = delka_datagramu;
      ip_dgram_fd_ptr->frag_len           = delka_datagramu;
      ip_dgram_fd_ptr->ttl                = 255;

      //protoze nasledujici hlavicka za touto zakladni bude "destination options", pouziju odpovidajici
            // nastaveni pole protokol
      ip_dgram_fd_ptr->protocol           = IpC_Procotol_Destination_Ext_Hdr;
      ip_dgram_fields_set (ip_packet_ptr, ip_dgram_fd_ptr);
```

```
                /*      =========TVORBA ROZSIRUJICI HLAVICKY========     */
//existuje struktura pro uchovavani zaznamu rozsirujicich hlavicek? Jestli ne, vytvor ji (jiz
        // implementovana fce).
if (!ipv6_extension_header_exists ((ip_dgram_fd_ptr)))
{
        (ip_dgram_fd_ptr)->ipv6_extension_hdr_info_ptr = (IpT_Ipv6_Extension_Headers_Data *)
                        ip_dgram_extension_header_struct_create ();
}

//vytvorim rozsirujici hlavicku destination options (jiz implementovana fce, upravena)
/*      Vstupni parametry:
        (IpC_Protocol_Mipv6_Proto_None – toto je poslední prenasena hlavicka, další neprenasim
        (care_of_address - jen pro potrebu splneni vstupnich parametru – promenna neni vyuzita.
                Tato adresa nahrazuje src v IP datagramu)
        RTSOLPR_ZPRAVA - vytvarim zpravu rtsolpr - identifikator pro prijemce (nesmi byt nula)
        Vystup:         ukazatel na rozsirujici hlavicku
*/

destination_hdr_ptr = (Ipv6T_Destination_Hdr_Info *) ipv6_destination_header_create
        (IpC_Protocol_Mipv6_Proto_None, care_of_address, RTSOLPR_ZPRAVA);

//Naplnim hlavicku daty. RtsolPr zprava bude nest tyto informace:
// current_bss_id       =               aktualni bss id pripojene AP (identifikator)
// nap_mac              =               MAC adresa pristiho AP (ke kteremu si zadam info o HO)
// code                 =               vyznam v RFC 5268

// Ziskani nap_mac
my_nap_mac = nar_mac;

// prirazeni hodnot
destination_hdr_ptr->data.current_bss_id = bss_id;              //ziskano z ICI od wlan_mac

// (hodnota je sem zapisovana z L2 primo, predpoklad:V okamziku zapisu do dh_data je jiz pro MN
        // tato struktura inicializovana)
destination_hdr_ptr->data.nap_mac = my_nap_mac;
destination_hdr_ptr->data.code = 0;                     //nastaveno napevno (momentalne jedina
        // definovana moznost, v RFC jich je vice)

// prirazeni hodnot
destination_hdr_ptr->data->current_bss_id = bss_id;             //ziskano z ICI od wlan_mac
destination_hdr_ptr->data->nap_mac = nap_mac;
//code nastaveno napevno (momentalne jedina definovana moznost, v RFC jich je vice)
destination_hdr_ptr->data->code = 0;

//zaradim tuto hlavicku na zacatek listu, ktery spravuje zaznam o pouzitych rozsirujicich hlavickach
op_prg_list_insert (ipv6_extension_header_list_get(ip_dgram_fd_ptr), destination_hdr_ptr ,
        OPC_LISTPOS_HEAD);
```

```
        //aktualizace poli diagramu
        op_pk_nfd_access (ip_packet_ptr, "fields", &ip_dgram_fd_ptr);

        //aktualizuji statistiku o odeslanych ridicich zprávách
        mipv6_ctrl_traffic_sent_stat_update (module_data_ptr, ip_packet_ptr);

        //odeslani datagramu do modulu IP
        op_pk_deliver (ip_packet_ptr, own_mod_objid, 0);

/* POZNÁMKA: Odesilani paketu do vzdaleneho modulu (zde chci odeslat paket z uzlu MN do uzlu PAR)
je koplikovanejsi zalezitost. Prikazem op_pk_deliver (Packet* paket, Objid modul_objid, int
index_vzdaleneho_prichoziho_streamu) dorucim paket do modulu IP. Tento provede smerovani paketu
v procesnim modelu ip_rte_central_cpu (pouzivam smerovani Central Processing) a odesle jej k PARu
(muze ho odeslat i na vyssi vrstvu, je-li to potreba!). Komunikacni cesta MN-PAR vypada nejak takto:
MN - IP(zde zpracovani funkcemi ip_rte_packet_arrival() a mipv6_sup_mobile_ipv6_packet_process() a
ip_rte_packet_send()) - putovani paketu siti - IP(paket prisel do IP z nizsi vrstvy, je zpracovan stejnymi
funkcemi jako u odesilatele (MN), nyni ale s pravidly pro uzel AP)-PAR.

Pri navrhu novych zprav je !!!NUTNE!!! overit pruchodnost teto zpravy popsanymi funkcemi, jinak se muze
stat, ze bude paket zahozen jeste u odesilatele (MN) na vrstve IP. Jedna se zdlouhavou cinnost :-( Blize
ke smerovani paketu v DP.        */

}

else
{
        //jestli nejsou splneny vsechny podminky, nemuzu zahajit vysilani RtSolPr zprávy
        FOUT;
}

FOUT;
}
```

Návrh využívá některých definovaných funkcí v OPNETu. Například vytváření IPv6 datagramu je svěřeno funkci ip_dgram_create(). Vytvoření diagram je pak naplněn zdrojovou, cílovou IPv6 adresou a je naplněna také informace o délce diagramu, pole TTL a pole následující hlavičky. Právě tato hlavička umožňuje přenést další informace od zdroje k cíli. Zvolena byla hlavička DH, které také odpovídá identifikace v poli následující hlavičky IpC_Procotol_Destination_Ext_Hdr (což je makro pro číslo). Vytváření rozšiřující hlavičky je pak detailně popsáno přímo v komentáři ke zdrojovému kódu. Pro přenos DH bylo nutné upravit funkce definované v *ipv6_extension_headers.ex.c*, jak již bylo popsáno dříve.

9.2.2 Modifikace funkcí zpracovávající rozšiřující hlavičky IPv6 paketů

V souvislosti s modifikací DH bylo nutno pozměnit i kód v podpůrné knihovně *ipv6_extension_headers_sup.ex.c*, který se stará o vytváření, zpracovávání a ničení rozšiřujících hlaviček paketu. Byla pozměněna funkce pro vytváření DH. Změna spočívá v přidání

identifikátoru do vstupních parametrů funkce ipv6_destination_header_create() v hlavičkovém souboru *ipv6_extension _headers_sup.h*. Nová definice funkce vypadá takto:

```
Ipv6T_Destination_Hdr_Info*        ipv6_destination_header_create (IpT_Protocol_Type next_hdr,
InetT_Address address, int identifikator);
```

Identifikátor slouží k identifikaci typu přenášené zprávy touto hlavičkou. DH mohu přenášet zprávu RtSolPr a PrRtAdv. Samotná funkce byla také upravena v podpůrné knihovně *ipv6_extension_headers_sup.ex.c* takto:

```
Ipv6T_Destination_Hdr_Info*
ipv6_destination_header_create (IpT_Protocol_Type next_hdr, InetT_Address address, int identifikator)
{
        //..predchazi puvodni kod
        destination_hdr_ptr->identifikator = identifikator;

        //pri vytvareni RtSolPr zpravy nechci prenaset home_adress, jako pro ostatni zprávy prenasene
        //pomoci DH
        if (  (destination_hdr_ptr->identifikator != RTSOLPR_ZPRAVA  ) &&
                ( destination_hdr_ptr->identifikator != PRRTADV_ZPRAVA)  )
        {
                destination_hdr_ptr->home_address = inet_address_copy (address);
        }
        //nasleduje puvodni kod

}
```

Byla také pozměněna funkce ipv6_destination_header_process(), která zpracovává rozšiřující hlavičku „destination“. U paketů s rozšiřující DH se tato hlavička při příjmu ničí. Toto rozhodně nemůžeme dopustit, protože DH nese zprávu RtSolPr a PrRtAdv. Přidáme tedy kód, který zajistí předání celého paketu s touto hlavičkou do procesního modelu *mipv6_mgr*, kde se bude paket dále zpracovávat. Přidáme vyjímku do funkce ipv6_destination_header_process(), která zajistí rozdílné zpracování standardní DH a DH nesoucí RtSolPr zprávu a PrRtAdv zprávu. Kód bude vypadat takto:

```
//…predchazi puvodni kod funkce ipv6_destination_header_process()

// vezmu hlavicku z paketu a uložím ji do ukazatele destination_hdr_ptr
destination_hdr_ptr = (Ipv6T_Destination_Hdr_Info *) op_prg_list_access (ipv6_extension_header_list_get
(*pkt_fields_pptr), OPC_LISTPOS_HEAD);

//paklize DH hlavickou neprenasim RtSolPr nebo PrRtAdv zpravu, je zpracovani puvodni
if ((destination_hdr_ptr->identifikator != RTSOLPR_ZPRAVA)          &&
        (destination_hdr_ptr->identifikator != PRRTADV_ZPRAVA))
{       //byla zmenena pouze podmínka tak, aby pri prijmu RtSolPr zprávy nedoslo ke zniceni DH
        // v tomto bloku se nachazi puvodni kod
```

```
        // v tomto bloku dochazi k nahrazeni zdrojove adresy paketu adresou z DH...
        // Po vykonani kodu předám paket zpet do IP modulu
FRET (OPC_TRUE);
}

//jestli se naopak RtSolPr zprava prenasi, preposlu paket do modelu mipv_mgr
else if (  destination_hdr_ptr->identifikator == RTSOLPR_ZPRAVA  )
{
        // zjistim "handle" na mipv6_mgr procesni model
        // handle se bude dat zjistit z dat uzlu (module_data)
        // Predam vykonavani programu do procesniho modelu mipv6_mgr
        // Pomoci op_pro_invoke() budu sdilet paket (pkptr)
        // handle na mipv6_mgr model je ulozen pri inicializaci mobility
                op_pro_invoke (module_data_ptr->mipv6_info_ptr->mipv6_prohandle, pkptr);

        //  Dam vedet nadrazene funkci, ze DH s RtSolPr zpravou jsem jiz zpracoval a neni proto treba ji
        // vracet do ip modulu
FRET (OPC_FALSE);
}

else if (destination_hdr_ptr->identifikator == PRRTADV_ZPRAVA)
{
        // zjistim "handle" na mipv6_mgr procesni model
        // handle se bude dat zjistit z dat uzlu (module_data)
        // predpoklad osetrim podminkou
        if (module_data_ptr->mipv6_info_ptr != OPC_NIL)
        {
                // Predam vykonavani programu do procesniho modelu mipv6_mgr.
                // Pomoci op_pro_invoke() budu sdilet paket (pkptr)
                // handle na mipv6_mgr model je ulozen pri inicializaci mobility
                op_pro_invoke (module_data_ptr->mipv6_info_ptr->mipv6_prohandle, pkptr);
        }

// dam vedet nadrazene funkci, ze DH s PrRtAdv zpravou jsem jiz zpracoval a neni proto treba ji vracet
        // ip modulu
FRET (OPC_FALSE);
}
```

Je také potřeba pozměnit definici samotné funkce ipv6_destination_header_process(), protože chceme zpracovat DH se zprávami RtSolPr a PrRtAdv podobně jako rozšiřující hlavičku „mobility“. Tedy tak, že přijaté IPv6 pakety s rozšiřující hlavičkou DH nesoucí zprávu RtSolPr respektive PrRtAdv[10] budou přeposílány do procesního modelu *mipv6_mgr*. Úprava spočívá

[10] Příjem takového paketu zpracovává funkce ipv6_extension_header_process() ve spolupráci s funkcí ipv6_destination_header_process(), která nahlédne do hlavičky a v případě, že nese RtSolPr, uloží hodnoty této zprávy a vyvolá obsloužení takovéhoto paketu v modelu mipv6_mgr. Samotná DH se do tohoto modelu již

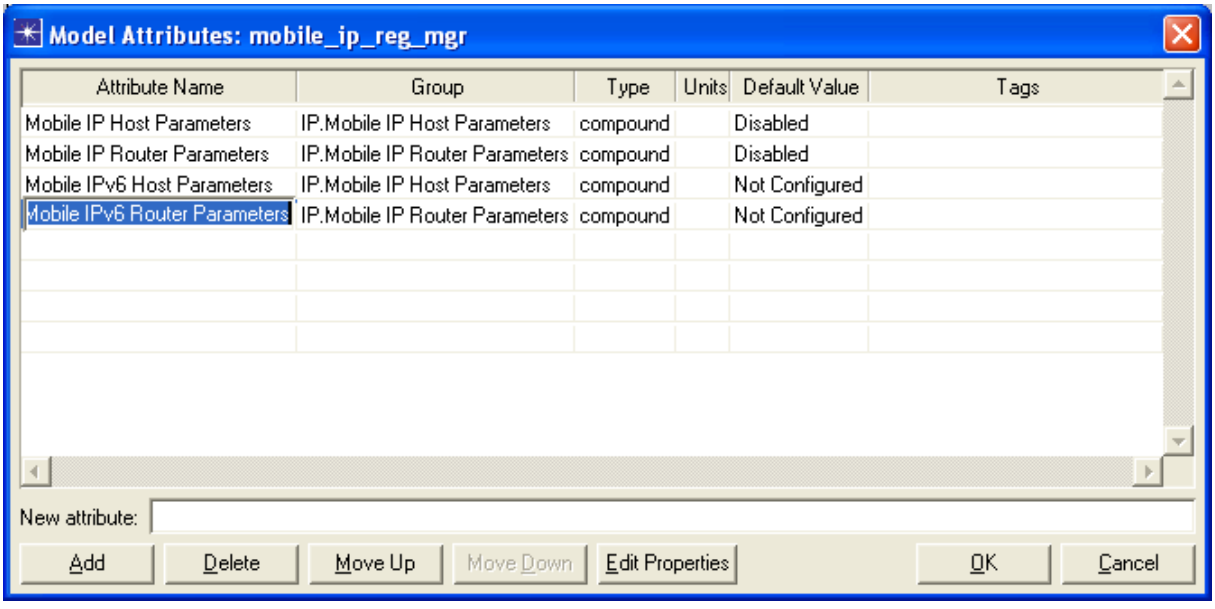
v přidání vstupních parametrů funkce a přidání výstupního parametru. Nová definice vypadá takto:

Boolean ipv6_destination_header_process (IpT_Dgram_Fields** pkt_fields_pptr, IpT_Rte_Module_Data* module_data_ptr, Packet *pkptr, IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr)

Díky výstupnímu parametru typu Boolean předáme funkci ipv6_extension_header_process() informaci o tom, že nechceme paket s DH s RtSolPr respektive PrRtAdv zprávou předat zpět do IP modulu (implementace je vidět na řádku 189 v *ipv6_extension_headers_sup.ex.c*). Vstupní parametry jsou podobné jako pro zpracování ipv6_mobility_header_process().

9.2.3 Modifikace podpory mobility ze strany routerů FN1 a FN2

Následuje přidání podpory mobility pro routery FN1 a FN2 v simulaci. Tyto routery budou plnit funkce PAR resp. NAR. Ve výchozím stavu můžeme u těchto routerů nastavit, aby plnily funkci HA, což ale nechceme. Vytvoříme novou položku v modulu *mobile_ip/Interfaces/Model Atributes* routeru. Viz obr. 9.2. Editujeme „Mobile IPv6 Router Parameters“ a následně „Interface Type“. Za položku „Home Agent“ přidáme novou položku „AP“ a hodnotu 6. Zbývá ještě doplnit výčtový typ uzlů v hlavičkovém souboru *mipv6_defs.h*. Jedná se o datový typ Mipv6T_Node_Type a doplníme do něj Fmipv6_AP = 6. Tento datový typ byl již popsán a je zařazen v Tab. 8.1. Nakonec vybereme pro routery FN1 a FN2 (editací jejich atributů v simulaci sítě) v *IP/Mobile IP Router Parameters/Mobile IPv6 Parameters/IF1/Interface Type* = AP.



Obr. 9.2: Přidání nových atributů pro zajištění podpory FMIPv6 u AP

nepřenáší. Díky oštření ipv6_extension_header_process() funkce se již paket dále nepředává zpět do IP modulu, celkově bude vyhodnocen v modelu mipv6_mgr.)

Nyní jsme zajistili, že routery FN1 a FN2 budou používat model *mipv6_mgr*. Toto je velmi důležité. Používání modelů lze ověřit v debuggeru. Při spuštění simulace FN1 a FN2 načtou (každý vlastní) model *mipv6_mgr* a proběhne inicializace a čtení parametrů. Schéma je uvedeno na Obr. 8.4.

9.2.4 Modifikace funkcí v podpůrné knihovně mipv6_sup.ex.c

Nyní již je zajištěná podpora pro práci s DH a můžeme se pustit do kontroly a modifikace kódu zajišťující odesílání resp. příjem paketu s DH. Při odesílání paketu s DH je tento předán do *mipv6_sup.ex.c* podpůrné knihovny a tam je vyhodnocen funkcí *mipv6_sup_mobile_ipv6_packet_process*, jak je naznačeno na Obr. 8.10. Kód v této funkci je pozměněn tak, aby nedocházelo při odesílání DH s RtSolPr respektive PrRtAdv zprávou k provádění nevhodných částí kódu. Upraven je do následující podoby:

```
else if ( ip_rte_node_is_mipv6_mobile_node (iprmd_ptr) )
{
        // Odesilam zpravu RtSolPr prostrednictvim DH z uzlu MN.
        // Rozhodne nechci tento paket vyhodnocovat jiz implementovanymi funkcemi pro MN,
        // proto pridam vyjimku pro RtSolPr zpravu (if (RTSOLPR) {muj kod} - else if {puvodni kod}).

        prenasim_dh = OPC_FALSE;
        // Zjisti, kolik je pripojenych hlavicek k tomuto paketu.
        pocet_hlavicek = ipv6_extension_header_list_size_get (*pk_fd_pptr);

        // Zjisti typ prenasene rozsirujici hlavicky.
        typ_hlavicky = (*pk_fd_pptr)->protocol;

        // Nejprve musim rozsirujici hlavicku ziskat z paketu
        if (pocet_hlavicek != 0 && typ_hlavicky == IpC_Procotol_Destination_Ext_Hdr)
        {
                extension_hdr_ptr = (Ipv6T_Destination_Hdr_Info *) op_prg_list_access
                        (ipv6_extension_header_list_get (*pk_fd_pptr), OPC_LISTPOS_HEAD);
                if (  (extension_hdr_ptr)->identifikator == RTSOLPR_ZPRAVA  )
                {
                        prenasim_dh = OPC_TRUE;
                }
        }

        // Vyhodnotim, je-li prenasena zprava RtSolPr
        if (prenasim_dh == OPC_TRUE)
        {
                // dalsi kod sem pridavat neni potreba
                // postacuje, ze jsme vyjimkou upravili zpracovani RtSolPr zprávy
        }
```

9.2.5 Zpracování zprávy

Samotné zpracování zpráv RtSolPr a PrRtAdv je koncipováno tak, aby příchozí paket s těmito zprávami byl odeslán do procesního modelu *mipv6_mgr*, kde bude adekvátně obsloužen. Protože původně kód v tomto stavu obsluhuje pouze rozšiřující hlavičky „mobility“ upravíme podmínky a přidáme kód pro zpracovávání hlaviček „destination“ (DH). Řešení popisuje následující kód:

```
/* Rozsiril jsem model mipv6_mgr o zpracovani paketu s rozsirujici hlavickou "Destination" */

else if ((IpT_Protocol_Type) pkt_fields_ptr->protocol == IpC_Procotol_Destination_Ext_Hdr)
{
	// do teto funkce jsou zatim predavany pouze pakety s DH se zpravou RtSolPr a PrRtAdv -
		// zajisteno na strane ipv6_destination_header_process()

	//odstranim DH ze seznamu k paketu pripojenych hlavicek, jiz ji nepotrebuji - obsah se mi ulozi do
		// destination_hdr_ptr
	destination_hdr_ptr = (Ipv6T_Destination_Hdr_Info *) op_prg_list_remove
		(ipv6_extension_header_list_get (pkt_fields_ptr), OPC_LISTPOS_HEAD);

	// aktualizuji statistiku prijatych ridicich paketu
	mipv6_ctrl_traffic_received_stat_update (module_data_ptr, pkptr);


			/* ===  VYHODNOTIM DATA Z RTSOLPR, ZPRACUJI A ODESLU ODPOVED === */

	// Poznamka: RtSolPr zpravou jsou aktualne prenaseny hodnoty code, nap_mac a current_bss_id.
	// Overim, ze zprava RtSolPr je zpracovavana v uzlu AP.
	if ((mipv6_node_type == Fmipv6_AP) && (destination_hdr_ptr->identifikator == RTSOLPR_ZPRAVA))
	{
		// Odpovidajici reakce na zpravu, dle code.
		switch (destination_hdr_ptr->data.code)
		{
		case 0:
		// V pripade, ze je code = 0, budu prenaset RtSolPr s informaci o HO,
		// tedy BSS ID, MAC adresu, IPv6 adresu a prefix site, do ktere bude MN vstupovat (NAR)

		/* 1] ZISKANI SPRAVNYCH HODNOT PRO PAKET ZPRAVY PRRTADV */
		// Zjistim bss_id prvku NAR.
		// Tato adresa bude rozhodujici pro spravne nastaveni dalsich parametru (napr nar_ipv6).
		// V simulaci musime dodrzet cislovani (BSS ID) nasledujicich AP tak, aby bylo vzdy o jedna
		// vetsi ve smeru postupu MN.
		nar_bss_id = ((destination_hdr_ptr->data.current_bss_id) + 1);

		// Zde mohu v budoucnu pridat podminky pro dalsi prvky AP v siti.
		// Momentalne zde mam definovanou IP adresu pouze pro jeden AP (FN2)
		switch (nar_bss_id)
		{
		case 3:
```

```
        // Budeme prenaset ip adresu NARu - zde je primo definovana (FN2 ze simulace)
        // Vytvorim IP adresu NARu (globalni i linkovou )a ulozim ji do destination_hdr_ptr.
        // Diky inet_address_create() mam zajisteno, ze se alokuje pamet pro tuto adresu.
        destination_hdr_ptr->data.nar_ipv6 = inet_address_create("2005:0:0:C:0:0:0:4",
        InetC_Addr_Family_v6);
        destination_hdr_ptr->data.nar_ipv6link = inet_address_create("FE80::0:0:0:4",
                InetC_Addr_Family_v6);

break;
}

/* 2] ULOZENI HODNOT DO DH_DATA_PTR (vlastni struktura) */

// Budu kopirovat data z destination_hdr_ptr do dh_data_ptr.
// Po zkopirovani potrebnych hodnot z DH, hlavicku znicim.
// Vytvorim tedy dh_data_ptr datoveho typu (Ipv6T_Destination_Hdr_Info *) v SV.
// Jedna se o dynamicky datovy typ (ukazatel), musim mu alokovat misto v pameti
dh_data_ptr = (Ipv6T_Destination_Hdr_Info *) op_prg_mem_alloc (sizeof
        (Ipv6T_Destination_Hdr_Info));

dh_data_ptr->data.code = destination_hdr_ptr->data.code;
dh_data_ptr->data.nap_mac = destination_hdr_ptr->data.nap_mac;
dh_data_ptr->data.current_bss_id = nar_bss_id;
// dale budu prenaset ve zprave informaci o prefixu site NAR
dh_data_ptr->data.nar_prefix = 64;
//je potreba take zkopirovat adresu NARU (linkovou i globalni)
//inet_address_copy() vraci strukturu InetT_Address a alokuje pro ni misto v dh_data_ptr
        // ->data.nar_ipv6
// tuto strukturu je nutne znicit, nebude-li potreba - pomoci inet_address_destroy()
dh_data_ptr->data.nar_ipv6 = inet_address_copy(destination_hdr_ptr->data.nar_ipv6);
dh_data_ptr->data.nar_ipv6link = inet_address_copy(destination_hdr_ptr
        ->data.nar_ipv6link);

/* 3] ZNICENI PUVODNI HLAVICKY destination_hdr_ptr            */
// Jelikoz jsou všechny potrebne udaje z RtSolPr zkopirovany, znicim DH
// Nejprve dealokovat vnitrni dynamicke struktury
inet_address_destroy (destination_hdr_ptr->home_address);
inet_address_destroy (destination_hdr_ptr->data.nar_ipv6);
inet_address_destroy (destination_hdr_ptr->data.nar_ipv6link);

// potom znicit vnejsi strukturu ukazatele
op_prg_mem_free (destination_hdr_ptr);
destination_hdr_ptr = OPC_NIL;

/* 4] VYTVORIME ZPRAVU PRRTADV */
// PrRtAdv je reakci na RtSolPr zpravu
// vstup: hodnoty ulozene v dh_data_ptr, cilova (src_addr) adresa, zdrojova (dest_addr)
```

```
                // adresa
        // struktura dh_data_ptr je znicena nize
        // cilova adresa je prevzata z prijateho paketu, zdrojova je linkova lokalni PARu
        prrtadv_source_addr = inet_address_create("FE80::0:0:0:3", InetC_Addr_Family_v6);
        prrtadv_dest_addr = inet_address_copy (pkt_fields_ptr->src_addr);

        fmipv6_prrtadv (dh_data_ptr, prrtadv_dest_addr, prrtadv_source_addr);
        // v tomto okamziku mam vytvořenou a odeslanou odpoved na RtSolPR…

        break;

        }

        /* 5] ZNICENI data_ptr, jiz neni potreba. */
        inet_address_destroy (dh_data_ptr->home_address);
        inet_address_destroy (dh_data_ptr->data.nar_ipv6);
        inet_address_destroy (dh_data_ptr->data.nar_ipv6link);
        op_prg_mem_free (dh_data_ptr);
        dh_data_ptr = OPC_NIL;

        inet_address_destroy (prrtadv_source_addr);
        inet_address_destroy (prrtadv_dest_addr);

}
```

/* PRIJEM PRRTADV ZPRAVY NA STRANE UZLU MN A JEJI VYHODNOCENI */

```
// Vykonavani programu se sem dostane pri prijmu paketu s hlavickou DH obsahujici zpravu
        // PrRtAdv z ipv6_extension_headers_sup.ex.c
// Pracovat mohu s celym prichozim paketem (IPv6 datagram vcetne rozsirujici hlavicky)
// PrRtAdv zpravu mohu zpracovavat pouze na strane MN - osetrim podminkou
else if ((mipv6_node_type == Mipv6C_Mobile_Node)          &&
        (destination_hdr_ptr->identifikator == PRRTADV_ZPRAVA))
{
        // Overim podminky nutne pro prijeti PrRtAdv dle specifikace

        // Zjistim NCoA
        nar_addr_range = inet_address_range_create (destination_hdr_ptr->data.nar_ipv6,
                inet_smask_from_length_create (destination_hdr_ptr->data.nar_prefix));
        // z vytvoreneho rozsahu vem prefix
        default_router_addr = inet_address_range_addr_get (&nar_addr_range);

        // overeni
        test = inet_rte_intf_addr_range_check (mobile_host_config_ptr->intf_info_ptr,
                default_router_addr, &ncoa_addr_range_ptr);
        // ulozeni NCoA
        *(module_data_ptr->mipv6_info_ptr->care_of_addr_ptr) =
                inet_address_range_addr_get_fast(ncoa_addr_range_ptr);
```

```
            // zniceni DH hlavicky destination_hdr_ptr
            inet_address_destroy (destination_hdr_ptr->home_address);
            inet_address_destroy (destination_hdr_ptr->data.nar_ipv6);
            inet_address_destroy (destination_hdr_ptr->data.nar_ipv6link);

            op_prg_mem_free (destination_hdr_ptr);
            destination_hdr_ptr = OPC_NIL;

        }

//znicim puvodni paket – jiz ho nepotrebuji, protože byl zpracovan
op_pk_destroy (pkptr);
}
```

Informace, které se šíří hlavičkou DH jsou přenášena v proměnné datového typu FMIPv6_data. Tyto hodnoty si můžeme představit jako pole data v rozšiřující hlavičce, které nám nese všechny informace obsažené v tomto datovém typu (struktuře). Jedná se o nově vytvořený datový typ, které lze nalézt v hlavičkovém souboru *fmipv6_support.h*. Je definovaný takto:

```
typedef struct Fmipv6_data
{
        // === pole v PrRtSol zprave ===
        //obsahuje aktualni bss_id
        short int               current_bss_id;

        //obsahuje L2 adresu nasledujiciho AP (datovy typ?)
        short int               nap_mac;
        short int               ident;

        // === pole v PrRtAdv ===
        //pole code informuje o vyznamu zprávy
        short int               code;

        //IPv6 adresa NARu
        InetT_Address           nar_ipv6;
        InetT_Address           nar_ipv6link;

        //informace o prefixu nasledujiciho NARu
        short int               nar_prefix;

} Fmipv6_data;
```

9.2.6 Zpráva PrRtAdv

Funkce pro vytváření této zprávy je implementována v bloku funkcí modelu *mipv6_mgr*. Funkce je volána ze stavu *IDLE mipv6_mgr* modelu po příjmu RtSolPr zprávy. Předpis funkce je takovýto: fmipv6_prrtadv (dh_data_ptr, prrtadv_dest_addr, prrtadv_source_addr); Do funkce tedy vstupují data z rozšiřující hlavičky a cílová a zdrojová adresa této zprávy. Funkce vytváří zprávu podobně, jako funkce pro vytváření RtSolPr, proto zde není uveden zdrojový kód. Je však v modelu

implementován. Rozdíl oproti RtSolPr zprávě je, že PrRtAdv přenáší bitově delší údaje, a proto je délka této rozšiřující hlavičky větší (256 bitů oproti 128 bitům).

Zpráva je přenášena z AP do MN a je při odesílání z AP vyhodnocována podobně jako RtSolPr zpráva. Před výstupem z AP do simulované sítě je zpracována funkcí ip_rte_packet_arrival() a odeslána na síťové rozhranní AP. U příjemce dojde k vyhodnocení paketu funkcemi v podpůrné knihovně *ipv6_extension_headers_sup.ex.c* a paket je dále přeposlán do procesního modelu *mipv6_mgr*, kde může být dále zpracována tato zpráva (nyní již je paket „fyzicky" přítomný u MN).

9.3 Návrh řešení dalších částí FMIPv6

Popsaný postup návrhu v této práci není kompletní, jelikož se nepodařilo v daném čase implementovat kompletní FMIPv6 protokol. V této části budou uvedeny stručně možnosti dalšího rozvoje protokolu.

Zpráva *FBU* bude poslána bezprostředně po přijetí zprávy PrRtAdv. Její tvar je podobný jako zpráva *BU* z protokolu MIPv6, jsou ale odlišné podmínky zpracování této zprávy. Zpráva bude vyhodnocována v AP (PAR) ne v domácím agentovi. Bude nutné tedy rozšířit podporu pro mobilitu v procesním modelu *mipv6_mgr* ve stavu *IDLE*. Také je nutno přidat podporu do *mipv6_sup.ex.c* pro zpracování této zprávy u MN a zkontrolovat zpracování rozšiřující hlavičky „mobility" u MN v *ipv6_extension_headers_sup.ex.c*. Je nutné přidat definici nového typu zprávy do *mipv6_defs.h* a v *ipv6_extension_headers_sup.ex.c* poupravit funkce pro vytváření, zpracování a ničení této nové zprávy. V případě potíží je nutné doplnit podporu ještě na úrovni směrování (modely pro směrování paketů jsou popsány v kapitole 8.4).

Zpráva *FBAck* jako reakce na *FBU* oznamuje MN, že bylo zaslání *FBU* úspěšné. AP musí bezprostředně po odeslání této zprávy začít přeposílat veškeré pakety směřující na aktuální CoA na adresu NCoA. Od této doby MN provádí „handover" mezi starou a novou cizí sítí. Pro AP to znamená, že musí být přidána podpora pro přeposílání paketů. Lze ji zajistit tak, že vložíme k rozšiřujícím hlavičkám hlavičku „směrování". Tato zajistí, že pakety budou muset na své cestě projít adresou uvedenou v této hlavičce (NARem) [20]. Pro MN je důležité, aby si ověřil, že *FBU* přišla ještě v momentě, kdy se vyskytoval v síti PAR. Kdyby tomu tak nebylo, musela by následující komunikace probíhat dle jiného schématu (reaktivní FMIPv6).

Zprávy *HI* a *Hack* mohou být dle nového RFC 5568 odesílány v rozšiřující hlavičce „mobility". Tyto zprávy slouží k přenesení adresy NCoA od následujícího AP (NAR) k aktuálnímu AP (PAR), a také k oznámení NARu, že od chvíle vyslání *HI* bude muset NAR ukládat příchozí komunikaci směřující do MN do té doby, než se MN přihlásí k síti NARu prostřednictvím zprávy. Jelikož NcoA se přenáší již zprávou PrRtAdv, není nutné implementaci této adresy do *HI* zavádět.

Pro implementaci těchto zpráv platí stejná metodika práce jako pro implementaci předchozích zpráv.

Bude nutné také vyřešit ukládání příchozích paketů v AP (NAR) do zásobníku (buffer). OPNET poskytuje pro vytváření zásobníku funkce. K nalezení jsou v [14].

Při příchodu MN do sítě prvku NAR, je tato událost detekována z linkové vrstvy signálem přerušení IPC_NAR_INTRPT_CODE (číslo kódu je 5302). Toto přerušení je již implementováno a šíří se z linkové vrstvy (*wlan_mac* modelu) do síťové vrstvy (*ip_dispatch*). V *ip_dispatch* ho tak stačí přeposlat dále do místa, odkod bude toto přerušení vyhodnocováno. Tedy do procesního modelu *mipv6_mn*. Po detekci tohoto signálu odešleme zprávu *UNA* a vyzvedneme si pakety uložené v zásobníku z prvku NAR. Dále již bude komunikace zajištěna protokolem MIPv6. FMIPv6 a MIPv6 mohou být implementovány ve stejných procesních modelech, protože protokol FMIPv6 pouze doplňuje podporu mobility v IPv6.

10 ZÁVĚR

Tato práce detailně analyzuje a diskutuje nad možnostmi, které poskytují dnešní IPv6 sítě v souvislosti s podporou mobility. Jsou zde popsány metody MIPv6, FMIPVv6, HMIPv6 a F-HMIPv6. Podpora mobility v IPv4 sítích musela být dodatečně implementována, kdežto v IPv6 sítích je tato podpora již standardně obsažena, a také umožňuje vývoj nových schémat předání handoveru, při cestování mobilního uživatele sítěmi.

Hlavním důvodem vzniku HMIPv6, byla nutnost snížit komunikaci mezi mobilním uživatelem a domácím agentem. Díky zavedením nových rozšíření v MIPv6, se podařilo tohoto cíle dosáhnout. HMIPv6 je popsána v kapitole 4. Detailní schéma výměny zpráv, je zobrazeno na obrázku 4.3.

Teprve až podpora rychlého handoveru (FMIPVv6), umožnila snížit zpoždění, při přechodu mobilního uživatele do cizí sítě. Tohoto je dosaženo, díky výměně informací mezi mobilním uživatelem a přípojným bodem sítě. Teoretické podklady pro činnost FMIPVv6 uvádí kapitola 3. Obrázky 3.2 a 3.3 ukazují výměnu zpráv v rámci FMIPVv6.

Nejvíce výhodnou, se jeví varianta F-HMIP, která zahrnuje podporu rychlého handoveru do hierarchické topologie. Má výhody obou předchozích metod. Snižuje zprávovou výměnu mezi mobilním uživatelem a domácím agentem a zároveň umožňuje přechod mezi sítěmi, v rámci jedoho MAPu (vysvětleno v kapitole 5). Obrázek 5.2 ukazuje, jak probíhá výměna zpráv mezi prvky topologie.

Standardní MIPv6 byla podrobena simulaci v prostředí OPNET. Topologie je znázorněna na obrázku 7.1 a splňuje zadání, které je popsáno v kapitole 7.1. Bylo ověřeno, že pomocí této techniky je možné komunikovat s mobilním účastníkem i při přechodu mezi různými sítěmi. Bohužel, při těchto přechodech (orig. „handover“), vzniká nebezpečí ztráty paketů. Ke ztrátě paketů dochází proto, že mobilní uživatel potřebuje určitou dobu k získání nové adresy v cizí síti a aktualizaci parametrů s komunikujícími stranami. Teoretické podklady pro činnost MIPv6 jsou uvedeny v kapitole 2. Obrázky 2.7 a 2.8 znázorňují výměnu zpráv. Výsledky simulace jsou shrnuty v kapitole 7.3. Jsou uvedeny grafické závislosti a komentovány jejich průběhy.

Součástí této práce je také analýza modelů zajišťující mobilitu v IPv6 síti v prostředí OPNET, za účelem návrhu vlastního protokolu odpovídající FMIPv6. Modely a zdrojové kódy těchto modelů, jejich funkce a struktura jsou popsány v kapitole 8. Závěrem lze snad jen říci, že implementace protokolu MIPv6 v OPNET Modeleru je kompletní a dle specifikace RFC 3775, ale je velmi obtížné se v ní zorientovat. Zpracování paketu zprávy MIPv6 protokolu prochází od MN k HA, CN resp. AP složitou cestou mezi modelem pro směrování paketu (*ip_rte_central cpu*) a funkcemi v podpůrných knihovnách *mipv6_sup.ex.c* a ipv6_*extension_headers_sup.ex.c*, až k modelům zpracovávající zprávy pro podporu mobility (*mipv6_mgr* a *mipv6_mn*). Po této

analýze následuje v kapitole 9 samotná implementace nového protokolu do prostředí OPNET, a sice protokol dle FMIPv6 (RFC 5568). Začátkem je třeba říci, že se nepodařilo implementovat celé řešení tohoto protokolu, jelikož se jedná o velmi časově náročnou činnost a nebylo možné tento návrh provést jednodužší cestou, protože by do budoucna nebylo možné v návrhu pokračovat a práce by tak neplnila svůj účel. Ve stávajícím návrhu je vyřešeno spouštění protokolu FMIPv6 pomocí událostí z linkové vrstvy uživatele MN. Jedná se o události detekce poklesu signálu na aktuálně připojeném AP a detekce připojení na nový AP. Implementace je provedena ve wlan_mac procesním modelu a v podpůrných knihovnách, ve funkcích, které detekují úroveň signálu AP. Je vyřešeno také šíření signálu přerušení spolu s dalšími informacemi z linkové do síťové vrstvy MN. Tento signál se šíří v několika stupních mezi různými procesními modely až do mipv6_mn, kde spouští FMIPv6. Zmíněné informace, které šíříme přerušením, obsahují mimo jiné i MAC adresu následujícího AP, která je ještě na úrovni linkové vrstvy RM OSI zjištěna z rámce „beacon“ šířící se MN od následujícího AP. Další úpravy jsou popsány v kapitole 9.1. Byla také vyřešena implementace zpráv RtSolPr a PrRtAdv. První ze zmiňovaných zpráv je přenášena mezi MN a AP. Je navržena jako IPv6 datagram s rozšiřující hlavičkou „destination header“ (DH). Druhá zpráva je přenášena opačným směrem a obsahuje informace o handoveru pro MN. Použití DH bylo zvoleno proto, že jsme využili podporu tohoto typu hlaviček v OPNETu. Avšak ani toto řešení nás neodpoutalo od nutnosti zavést podporu pro nově vytvořené zprávy na různých místech zdrojových kódů v OPNETu. Modifikace těchto kódů spolu s vysvětlením jednotlivých postupů lze nalézt v kapitole 9.2. Postupy při návrhu protokolu FMIPv6 mohou sloužit pro jeho rošíření (příklady uvádí kapitla 9.3), nebo mohou být využity pro jiné návrhy zprávové výměny, obecně protokolů na síťové vrstvě. Jelikož nebyla dokončena implementace FMIPv6, nemohla být provedena simulace na modelu využívající tento protokol. Nicméně by měly platit závěry k výhodám tohoto protokolu a nemělo by docházet ke ztrátám paketů při přechodu mezi sítěmi při cestování MN.

11 LITERATURA

[1] JOHNSON, D.; PERKINS, C.; ARKKO, J. *Mobility support in IPv6*. The Internet Engineering Force Task. RFC 3775, June 2004.

[2] KOODLI, R. *Mobile IPv6 Fast Handovers*. The Internet Engineering Force Task. RFC 5268, June 2008.

[3] SOLIMAN, H.; CASTELLUCIA, C.; ELMALKI, K. *Hierarchical Mobile IPv6 (HMIPv6) Mobility Management*. The Internet Engineering Force Task. RFC 5380, October 2008.

[4] SCHILLER, J. H. *Mobile Communications*. Great Britain: Addison-Wesley, 2003.

[5] JUNG, Heeyoung; SOLIMAN, Hesham; KOH, Seokyoo. *Fast Handover for HMIPv6*. Internet Engineering Task Force. Internet Draft, October 2005. <http://tools.ietf.org/id/draft-jung-mipshop-fhmipv6-00.txt.> [Citace: 20.10.009]

[6] JUNG, Heeyoung; KIM, Eunah; YI, Jongwha. *A Scheme for Supporting Fast Handover in Hierarchical Mobile IPv6 Networks*. ETRI Journal. Vol. 27, no. 6, pp. 798-801. December 2005.

[7] KOODLI, R. S., PERKINS, C. E. *Mobile Inter-networking with IPv6*: Concepts, Principles and Practices. Wiley-Interscience, 2007. 365 s. ISBN 978-0471681656.

[8] SATRAPA, Pavel. *Internetový protokol IPv6*. Praha: CZ.NIC, 2008. 357 s. ISBN 978-80-904248-0-7.

[9] *IPv6-Enabling the Mobile Internet.* NOKIA White Paper, 2000 <http://nds2.ir.nokia.com/NOKIA_COM_1/About_Nokia/Press/White_Papers/pdf_files/whitepawhi_IPv6_10878.pdf> [Citace: 5. 11. 2009].

[10] AURA, Tuomas. *Mobile IPv6 Security*. Proc. Security Protocols, 10th International Workshop, volume 2845 of LNCS, strany 215-228, Cambridge, UK, April 2002. <http://research.microsoft.com/en-us/um/people/tuomaura/publications/aura-protocols02.pdf> [Citace: 8. 11. 2009].

[11] PERKINS, C. *IP Mobility Support for IPv4.* Network Working Group. RFC 3344, August 2002.

[12] SKOŘEPA, Michal. Podpora mobility v IP sítích. Brno, 2007. 68 s. Vysoké učení technické v Brně. Vedoucí diplomové práce Ing. Karol Molnár, Ph. D.

[13] Enabling efficient and operational mobility in large heterogeneous IP networks. 2008. 280 s. <http://www.ipv6tf.org/pdf/enablebook.pdf> [Citace: 9. 12. 2009]. ISBN 978-84-691-0647-1.

[14] OPNET Technologies, Inc. *OPNET Modeler Release 14.5, Product documentation,* 2008.

[15] Prezentace: *Understanding Wireless LAN Model, Internals and Interfaces*. Session 1529. OPNET Technologies, Inc, 2008.

[16] ZEMAN OTTO. *Implementace simulačního modelu zjednodušené databáze diffserv-mib*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 61 s. Vedoucí diplomové práce Ing. Karol Molnár, Ph. D.

[17] BURDA KAREL. *Návrh, správa a bezpečnost počítačových sítí, přednáška č.8: Bezdrátové lokální sítě*. Brno: FEKT VUT v Brně. 2010.

[18] Prezentace: *Understanding IP Model Internals and Interfaces*. Session 1510. OPNET Technologies, Inc, 2008.

[19] PARK DANIEL S.; NJEDJOU ERIC. *L2 Triggers Optimized Mobile IPv6 Vertical Handover: The 802.11/GPRS Example*. Internet Draft. 2004

[20] *IPv6 Extension Headers Review and Considerations.* White Paper. Cisco Systems. 2006. 12 s.

12 SEZNAM ZKRATEK, VELIČIN A SYMBOLŮ

AP - přístupový bod

Binding cache - struktura pro uchovávání informací o vazbách na MN ve směrovačích

BU - Binding Update – zpráva používaná pro aktualizaci vazby v MIPv6

BU list - Binding Update list – struktura pro uchovávání informací o vazbách na MN v HA

CN - corespondent – korespondent

DH - Destination Header – typ rozšiřující hlavičky IPv6 datagramu

FMIPv6 - Fast Handovers for Mobile IPv6 – protokol rozšiřující MIPv6 (RFC 5568)

HA - home agent – domácí agent

Handle - zde nemá přímý český ekvivalent, významem se jedná oodkaz

HO - handover – předání komunikace mezi dvěma sítěmi

ICI - Interface Control Information – používá se při šíření informací spolu s přerušením

MAC - jedinečný identifikátor zařízení (rozhranní síťové karty) na linkové vrstvě OSI

MIPv6 - Mobile Ipv6 – protokol pro podporu mobility v IPv6 (RFC 3775)

MN - mobile node – mobilní uživatel

NAR - následující přístupový bod

PAR - předchozí přístupový bod

Objid - jedinečný identifikátor modulu v OPNETu

OPNET - program pro simulaci počítačových sítí